T H E S E S

présentées

pour l'obtention du titre

de

DOCTEUR DE L'UNIVERSITE

Spécialité : Informatique Appliquée

à

L'Université Pierre et Marie Curie
Paris 6

par

NGUYEN DUC Hieu

Sujet de la 1ère thèse : Réalisation d'un système de multiprogrammation pour un calculateur VARIAN 620/L-100 pilotant un microdensitomètre et des lignes de télétransmission.

Sujet de la 2ème thèse : La notion de"repère" dans les langages évolués, de Fortran à Algol 68, Pascal, et ADA.

Soutenues le 8 Mai 1981, devant la Commission composée de :

MM. B. ROBINET Président

R. DUPUY
D. TAUPIN Examinateurs
D. DACUNHA-CASTELLE Invité

T H E S E S

présentées

pour l'obtention du titre

de

DOCTEUR DE L'UNIVERSITE

Spécialité : Informatique Appliquée

à

L'Université Pierre et Marie Curie

Paris 6

par

NGUYEN DUC Hieu

Sujet de la 1ère thèse : Réalisation d'un système de multiprogrammation pour un calculateur VARIAN 620/L-100 pilotant un microdensitomètre et des lignes de télétransmission.

Sujet de la 2ème thèse : La notion de"repère" dans les langages évolués, de Fortran à Algol 68, Pascal, et ADA.

Soutenues le 8 Mai 1981, devant la Commission composée de :

MM. B. ROBINET Président

R. DUPUY
D. TAUPIN Examinateurs

D. DACUNHA-CASTELLE Invité

UNIV.
LEIDEN
BIBL.

REMERCIEMENTS

Je tiens à exprimer ma sincère reconnaissance à Monsieur le Professeur B. ROBINET qui a bien voulu présider ma thèse et me donner de bonnes directions.

J'exprime ma profonde gratitude à Monsieur le Directeur D. TAUPIN qui m'a accueilli au sein de son service, m'a donné un sujet intéressant, des suggestions importantes et des conditions de travail très favorables.

Je remercie vivement Madame J. NOGUES pour son amicale collaboration, les personnels du groupe système du CIRCE pour les conseils efficaces, Mesdames O. LANOUZIERE, C. DEFRANCE, Y. DUFOUR et H. THAISSEN pour la frappe et le tirage de ma thèse.

Je remercie sincèrement les personnels du Laboratoire de Physique des Solides qui m'ont tellement encouragé pendant mes études.

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41547318_0019

TABLE DES MATIERES

Pages

CHAPITRE 1 : INTRODUCTION

1. Présentation générale de l'installation VARIAN . . . 1
1.1. Processeur 1
1.2. Mémoire centrale 2
1.3. Unités périphériques 2
2. Jeu d'instructions du langage d'assemblage du VARIAN 4
3. But du projet 5

CHAPITRE 2 : PREPARATION DE MODULES CHARGEABLES

1. Politique d'allocation des unités périphériques 7
1.1.Unités périphériques partageables et non-partageables. . 7
1.2.Allocation des unités périphériques non-partageables . . 7
2. Préparation d'un module chargeable 8
3. Configuration de la mémoire centrale 11

CHAPITRE 3 : GESTION DES OPERATIONS D'ENTREE-SORTIE PAR DES INTERRUPTIONS ELEMENTAIRES

1. Aspect global 13
1.1.Apparition des interruptions 13
1.2.Principes généraux de la gestion des opérations d'entrée-sortie 18
1.3.Relations mutuelles 26
2. Traitement des lignes d'interruption 31
2.1.Ligne 1 : Microdensitomètre 31
2.2.Ligne 2 : Bande magnétique 34
2.3.Ligne 3 : Disque 39
2.4.Lignes 6 et 7 : Télétype central 41
2.5.Ligne 5 : Interface asynchrone multiple 43
2.6.Traitement d'interruption de l'horloge 52

Pages

CHAPITRE 4 : GESTION DES PROCESSUS CONCURRENTS

1. Processus concurrents 55
1.1. Caractéristiques des processus 55
1.2. Etat de processus 58
2. Fonctionnement de l'ordonnateur 59
3. Exécution des commandes 63
3.1. Chargement d'un programme en mémoire 63
3.2. Suppression de l'éxécution d'un programme 65
3.3. Mise en place des clés logiques 66
3.4. Suppression de l'éxécution d'une commande 66
4. Terminaison d'un programme 67
5. Aspect global du fonctionnement du calculateur 68

CHAPITRE 5 : GESTION DES FICHIERS SUR DISQUES

1. Organisation des fichiers 73
2. Création et suppression d'un fichier 74
3. Accès aux informations sur disque 76
3.1. Contrôle d'accès 76
3.2. Méthodes d'accès 77
4. Manipulation sur l'ensemble des fichiers 82

CHAPITRE 6 : FACILITIES D'EXPLOITATION DU CALCULATEUR

1. Facilités de l'exploitation quotidienne 86
1.1. Utilisation efficace de la bande de sauvegarde 86
1.2. Ensemble de messages d'observation et de diagnostic . . 87
1.3. Traitement des cas anormaux 88
1.4. Intervention directe sur les unités périphériques 90
2. Facilités de la mise au point d'un programme 90
3. Evaluation du système 91

ANNEXE 1 : GUIDE DE PROGRAMMATION 94
ANNEXE 2 : COMMANDES OPERATEUR 107
ANNEXE 3 : MESSAGES DU SYSTEME 109
BIBLIOGRAPHIE 111

C H A P I T R E I

INTRODUCTION

1 - PRESENTATION GENERALE DE L'INSTALLATION VARIAN

Le Service de Microdensitométrie du C.N.R.S. est destiné à traiter de façon quantitative les informations contenues dans les images photographiques . Dans ce but, il dispose d'un microdensitomètre piloté par un mini-ordinateur orienté vers le temps réel (VARIAN 620/L-100).

Ce mini-ordinateur est composé de 3 sections principales :

- un processeur
- une mémoire centrale
- des unités périphériques

Ce chapitre n'aborde que les détails d'installation concernant le projet. (Pour une information complète, se référer à la brochure "VARIAN 620-100 series" - 1972).

1.1. Processeur -

Le processeur contrôle toutes les opérations du calculateur. Il dispose de 4 registres d'opération nommés A, B, X et P. Chaque registre a une taille de 16 bits, i.e. celle d'un mot de la mémoire.

Le registre A joue le rôle d'un accumulateur. Il contient les résultats des opérations logiques et arithmétiques. Le registre B joue le rôle d'une extension du registre A utilisée pour les multiplication, division et les décalages. Les registres A et B peuvent servir aux instructions d'entrée-sortie. En outre, le registre B joue le rôle d'un registre d'index. Le registre X joue simplement le rôle d'un registre d'index.

Le registre P est le compteur ordinal. Il contient l'adresse de l'instruction qui suit directement l'instruction courante. Il est incrémenté régulièrement sauf en cas d'instructions de saut.

Les registres A, B et X sont accessibles directement par les instructions arithmétiques et logiques et de transfert.

En revanche, le registre P n'est accessible qu'indirectement à travers des instructions de saut et des instructions qui utilisent une adresse relative.

1.2. Mémoire centrale -

La mémoire centrale se compose de 16 K mots (1 K = 1024 mots). La taille d'un mot est 16 bits. L'unité adressable de la mémoire est le mot, ainsi, les mots de la mémoire sont numérotés de 0 à 037777 en octal. Toute la mémoire est directement adressable.

Le calculateur accepte les méthodes d'adressage suivantes : directe, indirecte (plusieurs niveaux), relative, indexée, extensive et immédiate.

1.3. Unités périphériques -

Le calculateur VARIAN pilote un microdensitomètre (Optronics P - 1000) et 2 lignes de télétransmission via des modems de 300 bauds (SAT 410) et de 1200 bauds. Il pilote en outre les unités périphériques suivantes : un télétype central (ASR 33) à 11 caractères par seconde, un télétype plus rapide (KSR 43) à 30 caractères par seconde, une imprimante à clavier (Decwriter 3) à 120 caractères par seconde , un dérouleur de bande (Kennedy 3110) et un disque amovible. Le disque, le dérouleur de bande, le télétype central et le microdensitomètre sont connectés directement à l'unité centrale. Nous les appelons "unités directes".

Les 2 modems, le télétype 300 bauds et l'imprimante à clavier* sont connectés à l'unité centrale via un interface. Nous les appelons "unités indirectes". Cet interface appelé "interface asynchrone multiple" comporte des interfaces de type ACIA (Asynchronous Communication Interface Adapter) pour chaque unité indirecte. Les deux lignes de télétransmission peuvent connecter le calculateur VARIAN avec l'ordinateur UNIVAC 1100, un ordinateur puissant du Centre de calcul Paris-Sud Informatique (PSI) (figure 1.1), ainsi qu'avec les ordinateurs IBM 370 (ou similaires) du Centre Interrégional de Calcul Electronique du CNRS (CIRCE). Pour des raisons logicielles, cette deuxième possibilité est actuellement peu utilisée.

---

* Dans le reste du texte, nous grouperons le télétype KSR 43 et l'imprimante à clavier Decwriter 3 sous le vocable "Imprimante".

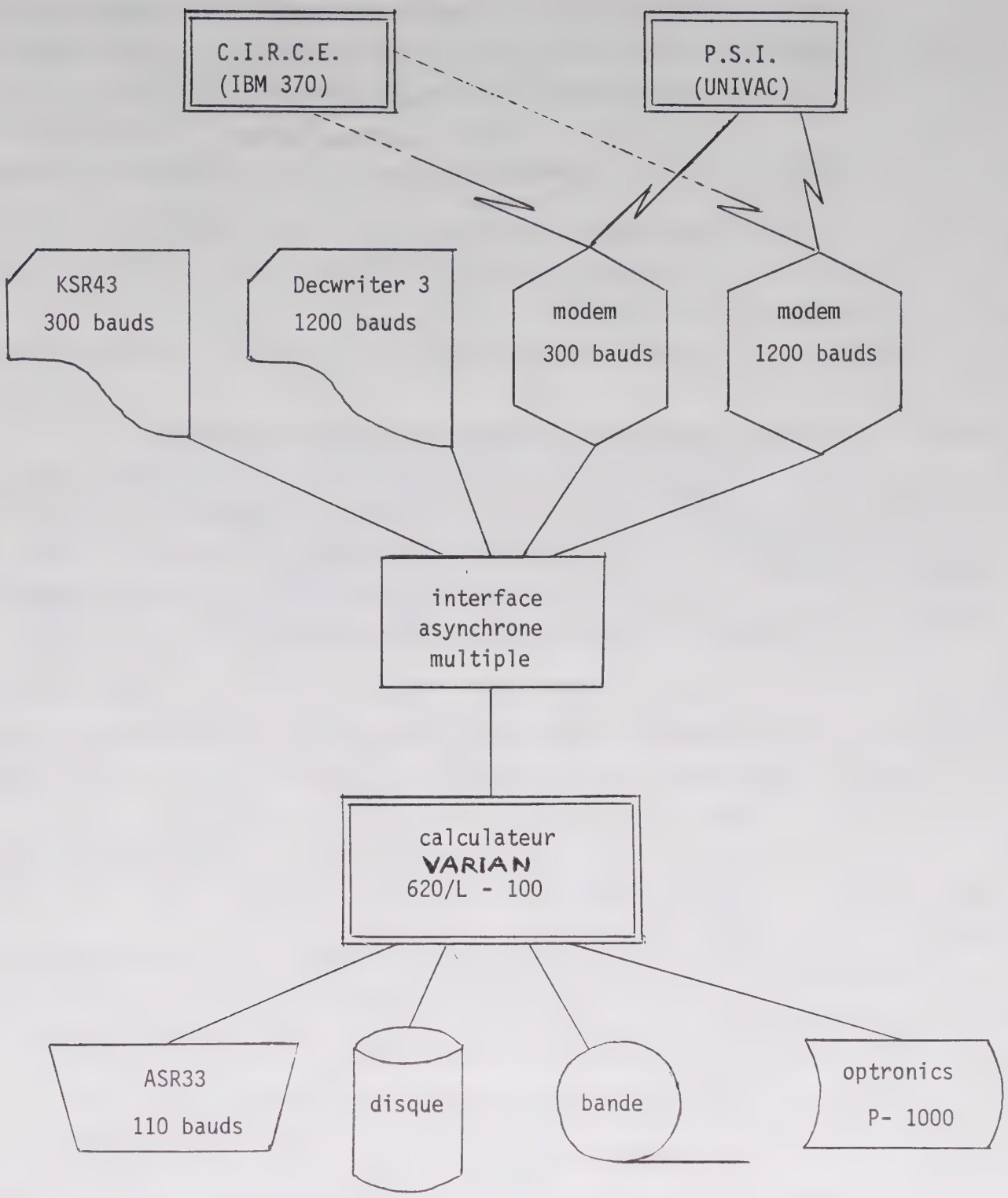


Figure 1.1. Installation VARIAN dans Service de Microdensitomètrie

Parmi les systèmes électroniques concernant les opérations d'entrée-sortie, on s'intéressera, plus particulièrement, à RTC (Real-Time Clock) et PIM (Priority Interrupt Module) qui ont été ajoutés dans le calculateur VARIAN et sont considérés comme des unités périphériques spéciales.

RTC s'occupe d'accumuler du temps durant le fonctionnement du calculateur et de provoquer périodiquement les interruptions du programme en cours.

PIM s'occupe d'enregistrer et de sélectionner les requêtes d'interruption en provenance des unités périphériques pendant des opérations d'entrée-sortie . Il permet aussi l'interruption du programme en cours.

Le fonctionnement en détail du RTC et PIM sera exposé dans le chapitre 3.

2 - JEU D'INSTRUCTIONS DU LANGAGE D'ASSEMBLAGE DU VARIAN

Chaque instruction se compose d'un code d'opération et éventuellement d'un opérande. Un opérande est une adresse de mémoire, ou une adresse conventionnelle d'une unité périphérique, ou bien une valeur.

La longueur d'une instruction est 1 mot ou 2 mots, suivant la méthode d'adressage et le groupe auquel elle appartient.

Le jeu d'instructions peut être divisé en 8 groupes principaux :

- Transferts d'information entre un mot de la mémoire et un registre(A,B,X).
- Décalage simple et décalage circulaire du contenu des registres (A, B).
- Modifications simples du contenu des registres (A, B, X)
- Instructions arithmétiques (addition, soustraction, multiplication, division) entre un mot de la mémoire et le registre accumulateur (le registre A).
- Instructions logiques (ET, OU inclusif, OU exclusif) entre un mot de la mémoire et le registre accumulateur A.
- Instructions d'entrée-sortie entre l'unité centrale et les unités périphériques.
- Instructions de saut conditionnel et inconditionnel.

Parmi les conditions possibles de saut, on s'intéresse à une condition spéciale : c'est l'état des clés sur le panneau du calculateur VARIAN. Grâce au changement manuel de l'état (en haut ou en bas) de ces clés, l'opérateur peut changer le déroulement du programme en cours si l'éxécution du programme rencontre une instruction de saut fonction de l'état des clés.

- Instructions de saut (conditionnel et inconditionnel) avec la sauvegarde du compteur ordinal.

Une instruction de ce groupe occupe 2 mots : le premier mot contient le code d'opération, le deuxième mot contient une adresse.

Exemple :
<adresse K> : code d'opération
<adresse K + 1> : une adresse I de la mémoire
Cette instruction s'éxécute en 2 phases :

- Sauvegarder le contenu actuel du compteur ordinal (l'adresse de l'instruction qui suit directement cette instruction de saut, i.e. l'adresse K + 2) dans le mot ayant l'adresse I.
- Remplacer le contenu du compteur ordinal par la valeur I + 1 et éxécuter la suite d'instructions à partir de l'adresse I + 1.

La sauvegarde du compteur ordinal sert à retrouver ultérieurement la séquence normale du programme qui a été modifiée par cette instruction de saut. Grâce à cette caractéristique, les instructions de ce groupe servent souvent à appeler des sous-programmes. Alors, elles sont appelées simplement les instructions de "saut avec retour".

On remarque qu'une instruction de saut avec retour modifie toujours un mot dans le sous-programme appelé pour y mettre l'adresse de retour au programme principal, aussi est-il impossible d'écrire des sous-programmes réentrants.

3 - BUT DU PROJET

Jusqu'à présent, le calculateur fonctionnait sous un système de monoprogrammation. Un programme est chargé en mémoire de façon fixe avec des adresses absolues. Chaque fois que le programme déclenche une opération d'entrée-sortie, le processeur est occupé pendant la durée de cette opération par des boucles d'attente de changement d'état de l'unité périphérique concernée. En conséquence, il n'y a qu'une opération d'entrée-sortie à la fois. De plus, quelques programmes d'application demandent un temps d'éxécution assez long, même de l'ordre des heures. Un programme pareil reporte consiérablement l'éxécution des autres programmes.

Le but de ce projet est de réaliser un système d'exploitation du calculateur VARIAN qui permet à l'unité centrale et à toutes les unités périphériques de pouvoir fonctionner de façon parallèle. Enfin, ce système doit permettre l'éxécution simultanée de 2 programmes indépendants coexistant en mémoire.

Pour atteindre ce but, il est nécessaire de résoudre les problèmes suivants :

- Politique d'allocation des ressources (mémoire centrale et unités périphériques) et préparation de modules chargeables dont les adresses sont translatables.

- Gestion des opérations d'entrée-sortie par des interruptions élémentaires. C'est la couche de base du système. (figure 1.2)

- Gestion des processus concurrents.

- Gestion des fichiers.

- Facilités d'exploitation.

Les solutions de ces problèmes seront exposées dans les chapitres qui suivent.

Le langage de programmation utilisé pour la réalisation du projet est le langage d'assemblage du calculateur VARIAN 620/L-100, modifié pour utiliser les possibilités d'assemblage "croisé" sur l'ordinateur UNIVAC 1100.

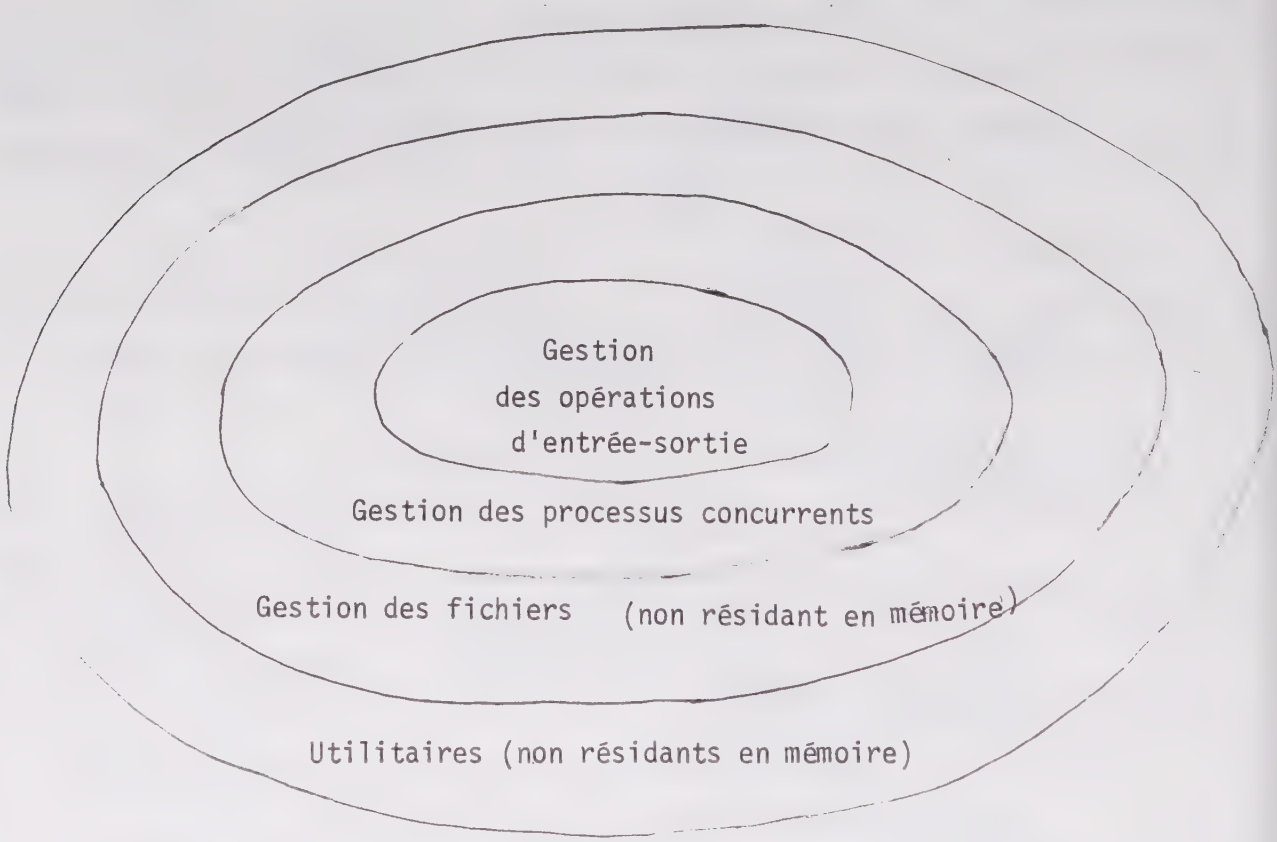


Figure 1.2. - Structure générale du système d'exploitation du calculateur VARIAN

C H A P I T R E II

PREPARATION DE MODULES CHARGEABLES

1 - POLITIQUE D'ALLOCATION DES UNITES PERIPHERIQUES

Sous le système de monoprogrammation, toutes les unités périphériques sont à la disposition du programme en cours. Or, si l'on veut que 2 programmes puissent s'éxécuter en même temps, il faut avoir une politique convenable d'allocation des unités périphériques afin d'éviter les conflits.

1.1. Unités périphériques partageables et non-partageables -

Les caractéristiques de fonctionnement du matériel de quelques unités périphériques, telles que le dérouleur de bande, le microdensitomètre, le modem, font que chaque unité n'est pas partageable entre les programmes concurrents.

En ce qui concerne les imprimantes KSR43 et Decwriter 3, on peut envisager d'en dédier une à chaque programme coexistant s'il en a besoin. Alors, chacune de ces deux imprimantes est aussi considérée comme non-partageable.

Seuls le télétype central (jouant le rôle de console d'opérateur)et le disque sont des unités périphériques partageables. Il n'y aura donc pas de problème d'allocation pour ces deux unités. Par contre, il est nécessaire de définir les principes pour l'allocation des autres unités.

1.2. Allocation des unités périphériques non-partageables -

Chaque programme a donc à sa disposition le télétype central et le disque. Pour avoir les autres unités périphériques non-partageables nécessaires dans l'éxécution d'un programme, il faut que ce programme déclare une liste de celles-ci. En ce qui concerne les imprimantes et/ou les modems, le programme ne précise pas encore telle imprimante (KSR 43 ou Decwriter 3) et/ou tel modem (300 bauds ou 1200 bauds). Au moment du chargement du programme en mémoire, le système désignera définitivement une imprimante et/ou un modem pour ce

programme. Ceci implique qu'un programme ne peut requérir les deux imprimantes à clavier et/ou les deux modems.

Les unités périphériques non-partageables qui sont allouées à un programme appartiennent à celui-ci pendant toute son éxécution.

2 - PREPARATION D'UN MODULE CHARGEABLE

Le but de cette étape est de créer un module dont les adresses sont translatables, prêt à être chargé en mémoire pour s'éxécuter. Cette étape est réalisée sur l'ordinateur UNIVAC.

Pour cela, on a mis au point une série de procédures (macro-instructions pour l'assembleur UNIVAC, portant les mêmes noms mnémoniques des instructions du VARIAN. Ces procédures permettent d'utiliser l'assembleur normalement prévu pour UNIVAC 1100, pour engendrer du code machine adapté au VARIAN 620/L-100 (se référer au document "Utilisation de l'assembleur SLEUTH II" de D. Taupin et J. Noguès, 1977).

Les noms des instructions sont spécifiques du VARIAN mais le langage d'assemblage maintenant a donc la syntaxe de l'assembleur UNIVAC. Ainsi, on peut compiler les programmes écrits pour VARIAN avec les mêmes facilités que n'importe quel programme écrit pour UNIVAC.

De plus, on ajoute sur l'UNIVAC un programme écrit réellement en langage d'assemblage UNIVAC. Après l'assemblage, ce programme devient un module objet type UNIVAC qui est appelé "loader" (figure 2.1).L'éditeur de liens de l'UNIVAC permet alors de créer un module éxécutable sur UNIVAC. Ce module se compose de deux parties : une partie d'instructions UNIVAC (le "loader") et une partie de données au sens de l'UNIVAC, un programme principal et ses sous-programmes de langage du VARIAN compilés séparément, avec la référence des symboles externes entre eux. L'éxécution de ce module dans l'UNIVAC crée un module chargeable pour le VARIAN sur une bande magnétique ou dans un fichier destiné à une transmission téléphonique.

Un module chargeable se compose de 3 parties :

- une partie appelée l'en-tête du programme contient les informations de description globale du programme; parmi elles, on trouve le nom du programme, l'adresse relative de début de son éxécution (par rapport au début du module), la taille mémoire et la liste des unités périphériques non-partageables nécessaire pour l'éxécution du programme.

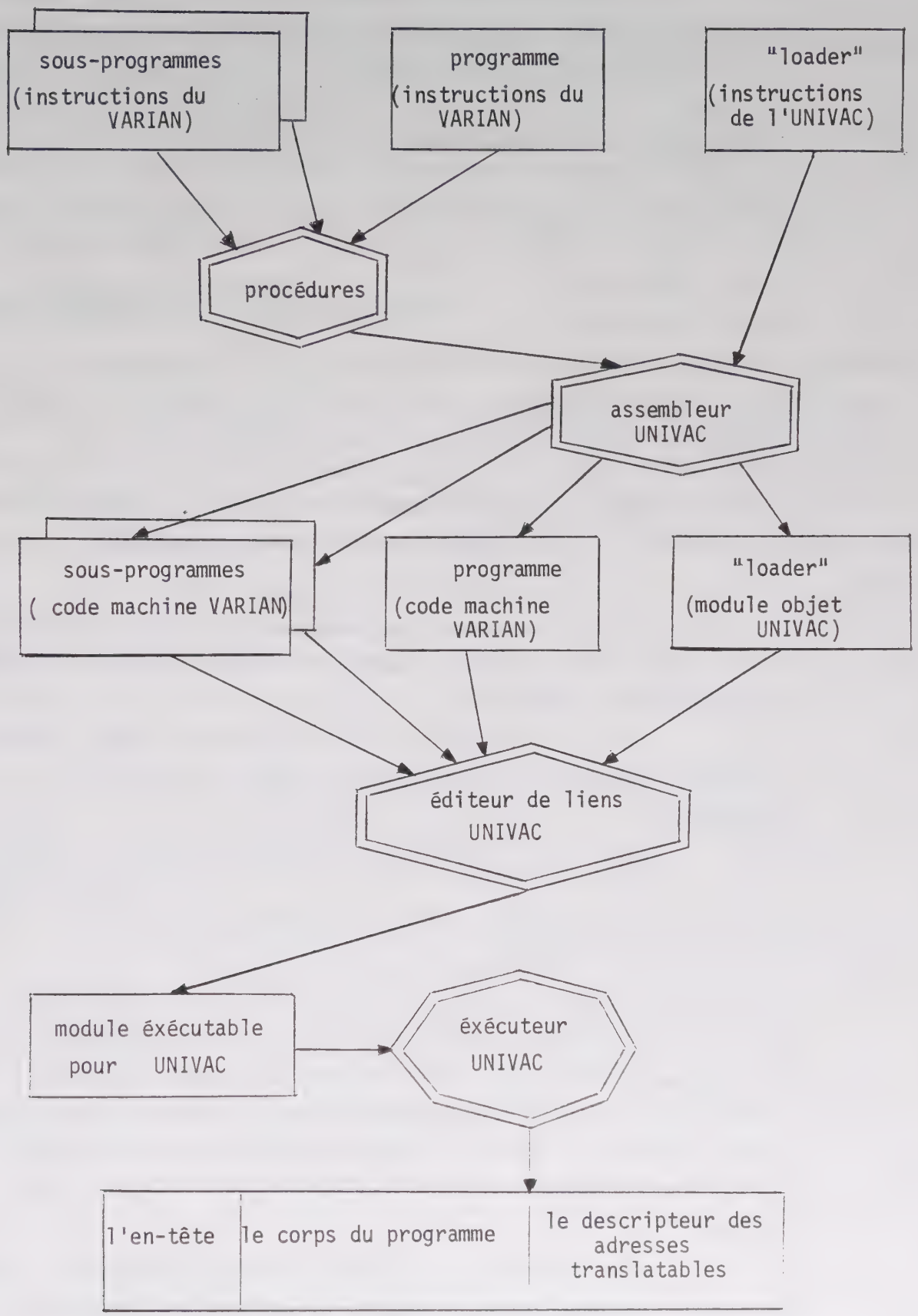


Figure.2.1 - Préparation d'un module chargeable pour le VARIAN

La liste des unités est décrite par un mot dont chaque bit représente conventionnellement une unité périphérique. Un bit qui vaut 1 signifie la demande de l'unité périphérique correspondante. En ce qui concerne les imprimantes et les modems, il faut réserver des bits supplémentaires pour indiquer l'unité d'imprimante ou de modem désignée ultérieurement par le système (figure 2.2.).

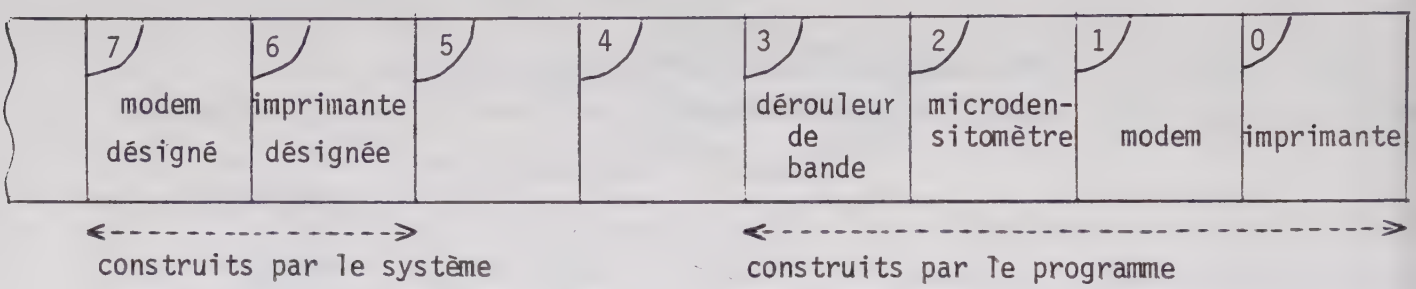


Figure 2.2. Description de la liste des unités périphériques non partageables pour un programme.

- Une partie appelée le "corps du programme" qui correspond à une suite d'instructions et de données reconnaissables par le calculateur VARIAN, avec des adresses translatables.

- Une partie appelée le "descripteur des adresses translatables du programme" qui donne les positions des adresses translatables dans le corps du programme.

Chaque bit de cette partie représente un mot du corps du programme. La longueur de cette partie dépend donc de la longueur du corps du programme. Un bit qui vaut 1 signifie que le mot correspondant dans la partie du corps du programme est une adresse relative.

On remarque que tous les derniers mots consécutifs de cette partie qui valent 0 sont inutiles, en conséquence, on peut les ignorer afin de diminuer la longueur de cette partie. Alors, un programme bien organisé (i.e. dont les parties concernant des adresses translatables sont rassemblées dans la première zone du corps) permet de minimiser la place mémoire nécessaire au moment du chargement et aussi d'économiser la place de stockage sur disque ainsi que le temps de télétransmission.

Ainsi, lorsque l'on a obtenu un module chargeable, l'étape de la préparation est achevée. La bande magnétique qui contient les modules chargeables

est transportée au Service de Microdensitométrie, et les modules choisis seront chargés en mémoire centrale du VARIAN pour y être éxécutés.

Dans le chapitre "Facilités d'exploitation", on verra que les modules chargeables peuvent aussi être stockés sur un disque de l'UNIVAC. Ensuite, ils seront transmis via une ligne de télétransmission au disque du VARIAN, grâce à un programme utilitaire.

3 - CONFIGURATION DE LA MEMOIRE CENTRALE -

La mémoire centrale est divisée en 2 zones fixes : une zone réservée pour la partie résidante du système et une zone réservée pour les programmes.

La zone du système occupe 2 parties séparées de la mémoire centrale qui s'appellent la partie A (partie basse des adresses) et la partie B (partie haute des adresses). La zone du système comprend les sous-programmes d'entrée-sortie, les procédures de la gestion des processus, les blocs de contrôle, etc...

Le reste de la mémoire est une zone continue prévue pour 2 programmes au maximum. Au moment du chargement, le corps et ensuite le descripteur des adresses translatables d'un programme y seront chargés. Suivant la position du programme en mémoire, on le numérote 2 ou 3 (le numéro 1 est réservé pour le système)(figure 2.3).

Lorsque les informations contenues dans le descripteur des adresses translatables ont été utilisées pour transformer les adresses concernées, cette partie devient inutile et il est alors possible de récupérer la zone mémoire qu'elle utilisait.

Ceci permet de gagner un peu de place mémoire pour le programme numéro 3 si ce programme est chargé après le programme numéro 2. Pour cela, le chargeur du VARIAN choisit tout d'abord la partie basse des adresses de mémoire (la position du programme numéro 2) pour charger un programme en mémoire quand la zone réservée pour les deux programmes est encore totalement vide (se référer au chapitre "Gestion des processus concurrents").

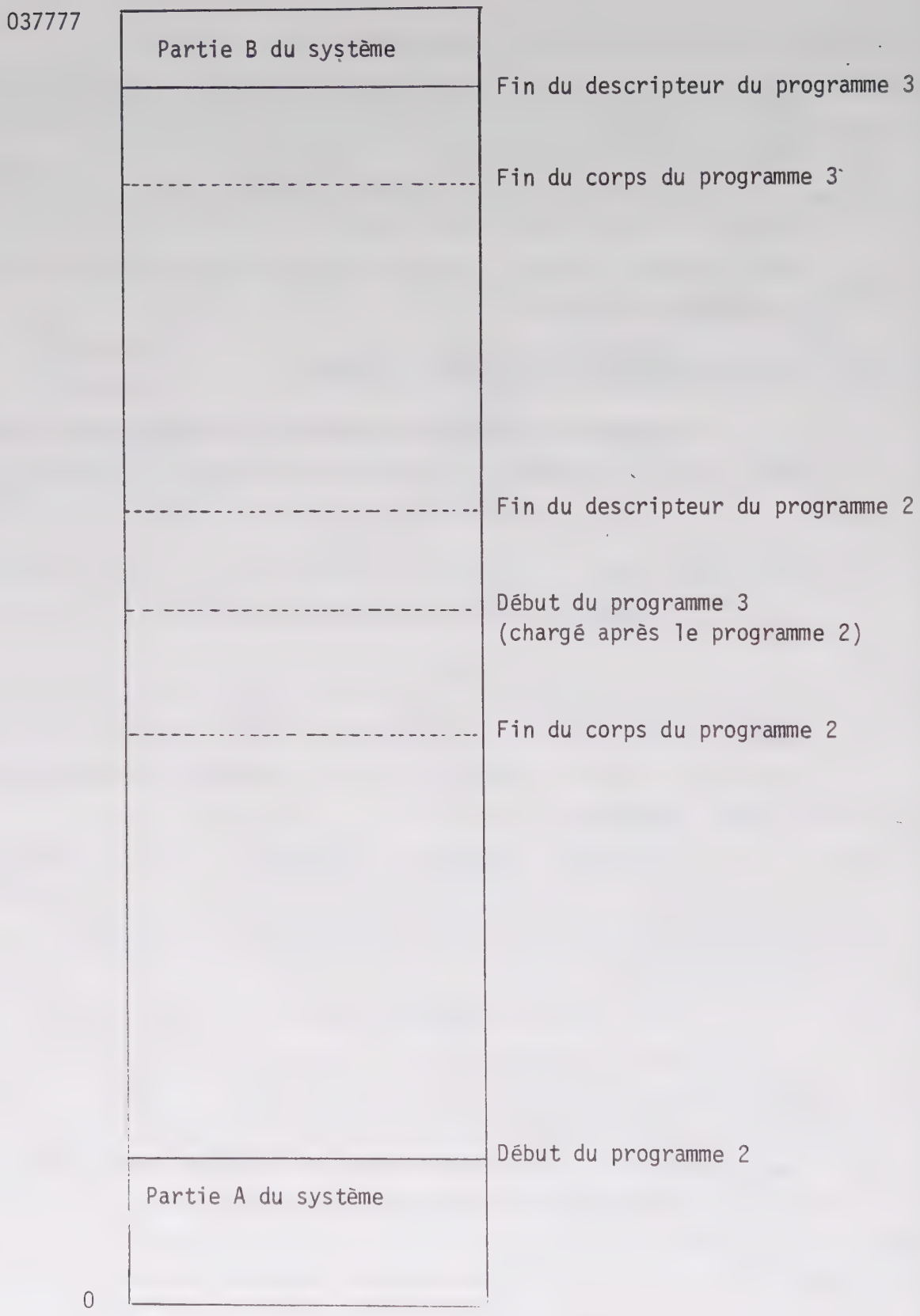


Figure 2.3. - Configuration de la mémoire centrale

C H A P I T R E III

GESTION DES OPERATIONS D'ENTREE-SORTIE PAR DES INTERRUPTIONS ELEMENTAIRES

1 - ASPECT GLOBAL

1.1. Apparition des interruptions -

Il y a les interruptions de l'horloge et les interruptions des unités périphériques. L'apparition des interruptions est observée en deux phases :

- phase 1 : une requête d'interruption est enregistrée ;
- phase 2 : cette requête est reconnue et le processus en cours d'exécution est interrompu provisoirement par le traitement de la requête d'interruption.

1.1.1. Apparition des interruptions de l'horloge

Le calculateur VARIAN dispose d'une carte électronique appelée "l'horloge de temps réel" (RTC) pour compter le temps (réel) pendant le fonctionnement de l'unité centrale. Le premier composant de l'horloge est un compteur interne.

Ce compteur s'incrémente régulièrement pendant le fonctionnement du calculateur et son débordement qui a lieu toutes les millisecondes produit une requête d'interruption. Le traitement de cette requête revient simplement à exécuter une instruction qui incrémente un compteur externe. Celui-ci est situé à une adresse fixe de la mémoire.

Le débordement du compteur externe produit un nouveau type de requête d'interruption de l'horloge. Quand cette requête est reconnue, le processus en cours d'exécution est interrompu et le processeur exécute une instruction à une adresse fixe. C'est une instruction "saut avec retour" qui pointe vers la partie de traitement d'interruption de l'horloge.

Désormais, on ne s'intéresse qu'à ce deuxième type d'interruption de l'horloge.

1.1.2. Apparition des interruptions des unités périphériques

Le VARIAN dispose d'une carte électronique appelée PIM pour enregistrer et sélectionner des requêtes d'interruption marquant un changement d'état des unités périphériques. Les unités sont gérées par des contrôleurs. Outre les lignes de "données", des lignes d'interruption relient ces contrôleurs à l'unité centrale.

Le PIM possède un registre de masque et un registre d'interruption. Chacun de ces registres a 8 bits qui correspondent aux 8 lignes d'interruption connectées aux contrôleurs des unités périphériques. Le registre de masque a pour but de masquer les lignes choisies. Le registre d'interruption a pour but d'enregistrer les requêtes d'interruption en provenance des lignes d'interruption.

Toutes les 950 nanosecondes, le PIM balaye le registre d'interruption pour détecter les requêtes des lignes dans l'ordre des priorités (câblées), et choisira la première requête rencontrée. L'ordre de priorité de ces lignes a été décidé en fonction des caractéristiques des opérations des unités périphériques connectées, et particulièrement en fonction du niveau d'urgence de traitement des interruptions.

Quand une requête est reconnue, le processus en cours d'exécution est interrompu et le processeur exécute une instruction à une adresse fixe liée à la ligne pour laquelle l'interruption est arrivée. C'est une instruction "saut avec retour" qui pointe vers la partie de traitement correspondante.

Une unité périphérique peut être connectée à plusieurs lignes, de même que plusieurs unités périphériques peuvent partager une ligne via un interface. L'établissement actuel des lignes d'interruption dans le calculateur VARIAN est donné par la figure 3.1..

1.1.3. Empêchement d'apparition des interruptions

Les requêtes d'interruption sont produites par l'horloge et les unités périphériques à des moments précis où leur état est changé selon un sens défini. L'enregistrement et la reconnaissance de ces requêtes sont à la charge des cartes RTC et PIM.

Cependant, RTC et PIM sont considérées comme des unités périphériques spéciales pour le calculateur ; par conséquent, on peut les influencer en utilisant des instructions du type entrée-sortie. Pendant des intervalles de temps choisis, on peut ignorer certaines requêtes d'interruption ou retarder la reconnaissance des requêtes, et même supprimer certaines requêtes en attente de traitement.

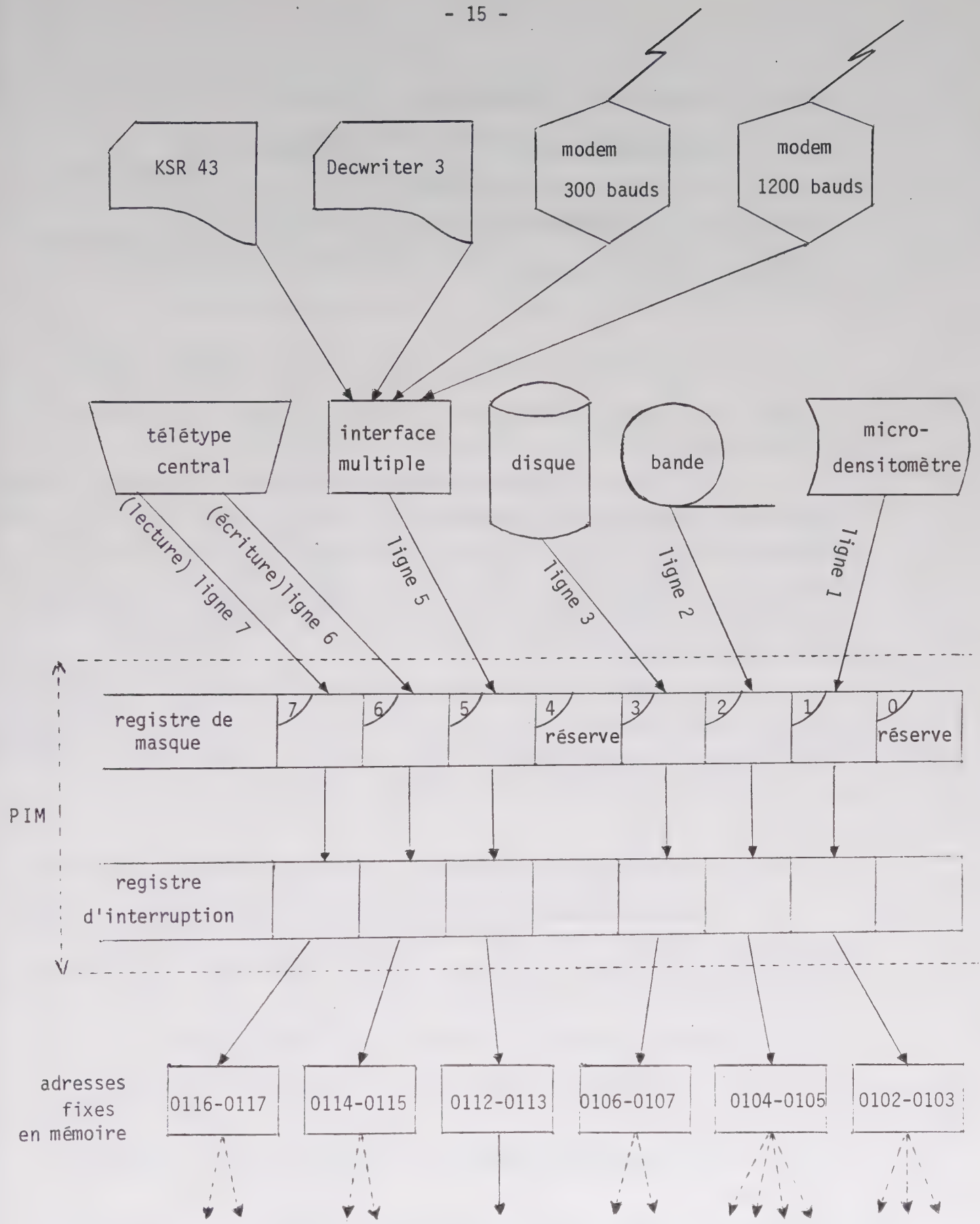


Figure 3.1. Lignes d'interruption des unités périphériques

1.1.3.1. Empêchement d'interruption de l'horloge

On s'intéresse aux possibilités suivantes :

- ignorer les deux types d'interruption de l'horloge (i.e., interdire RTC),
- autoriser les deux types d'interruption (i.e., autoriser RTC),
- choisir la valeur initiale du compteur externe pour avoir un rythme valable des interruptions de l'horloge.

1.1.3.2. Empêchement d'interruption des unités périphériques directes

Les unités périphériques directes sont : le dérouleur de bande, le microdensitomètre, le disque et le télétype central.

On utilise le registre de masque pour ignorer les requêtes d'interruption des lignes choisies. Le contenu de ce registre peut être modifié par des instructions. Chaque ligne d'interruption correspond à un bit du registre de masque. Si ce bit vaut 1, les requêtes d'interruption provenant de la ligne correspondante seront ignorées. Si ce bit vaut 0, alors, quand une requête d'interruption arrive, elle est enregistrée dans le registre d'interruption.

On retarde la reconnaissance des requêtes en attente de traitement dans le registre d'interruption par une instruction qui interdit le balayage du PIM sur ce registre. Ce balayage ne reprend qu'après une instruction spéciale qui l'autorise explicitement (voir la figure 3.2.).

1.1.3.3. Empêchement d'interruption des unités périphériques indirectes

Les unités périphériques indirectes sont les imprimantes KSR43 et Decwriter 3, les modems 300 bauds et 1.200 bauds. Chaque unité dispose d'un registre de contrôle (ACIA Control Register) qui établit une permission d'enregistrer des requêtes d'interruption au niveau local de cette unité. Il y a 3 sortes de permission suivant le contenu du registre de contrôle :

- enregistrer uniquement les requêtes d'interruption de sens "entrée" pour la lecture ;
- enregistrer uniquement les requêtes d'interruption de sens "sortie" pour l'écriture ;
- ignorer les requêtes dans les deux sens.

La permission peut être modifiée par des instructions.

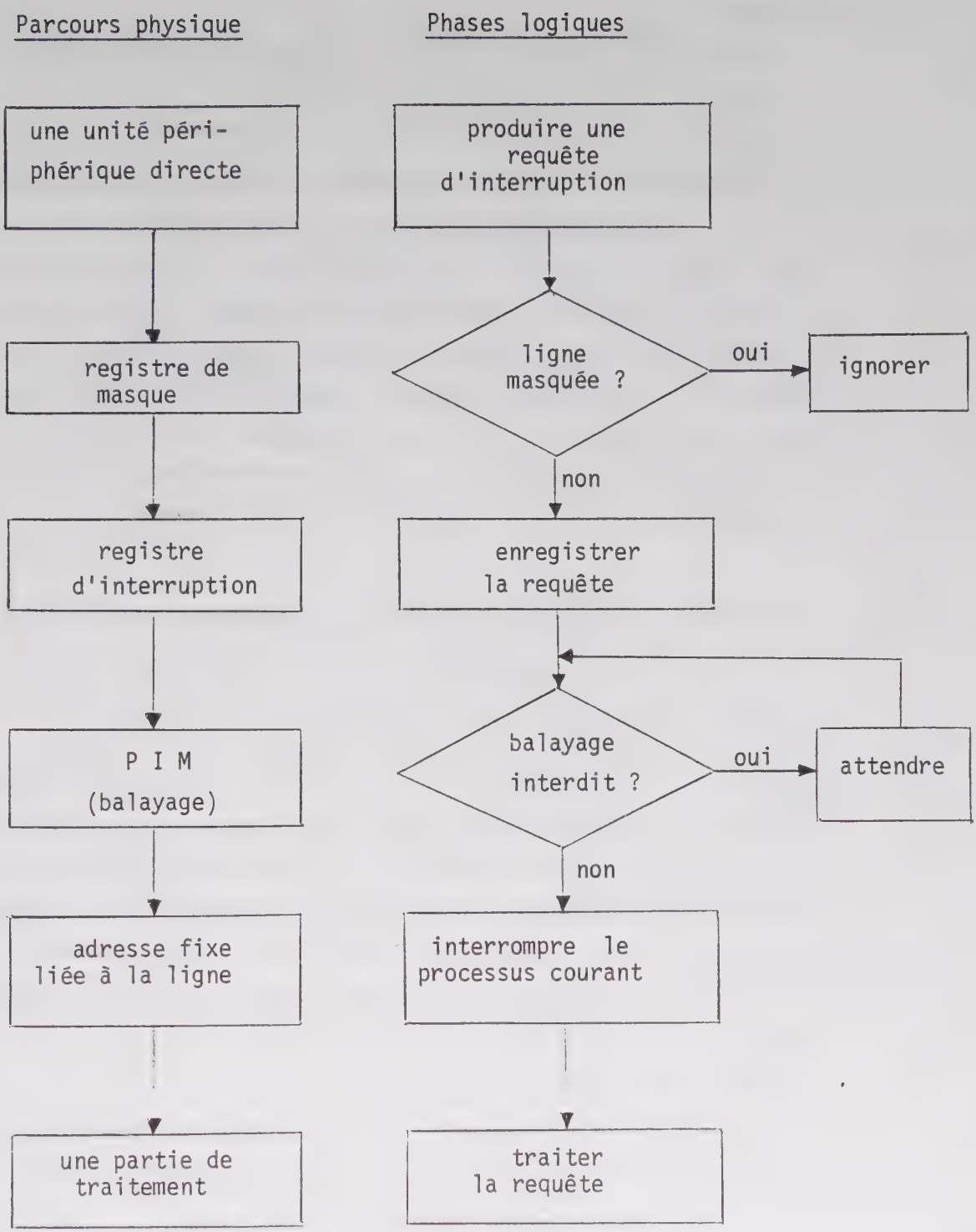


Figure 3.2. - L'apparition et l'empêchement d'interruption d'une unité périphérique directe

Une requête sera enregistrée dans le registre d'interruption du PIM si les deux conditions suivantes sont satisfaites lorsque cette requête apparaît :

- aucune autre requête n'est en attente sur la ligne d'interruption numéro 5 (la ligne liée à l'interface) ;
- la ligne 5 n'est pas masquée par le registre de masque.(figure 3.3.)

La requête ne sera prise en compte que lors du balayage du PIM.

En revanche, une requête qui ne satisfait pas au moins à une condition ci-dessus reste dans le registre d'état de l'unité périphérique correspondante pour attendre un traitement sans provoquer aucune interrruption au niveau du VARIAN. Ceci s'explique par la structure de la ligne d'interruption : son niveau est la réunion de toutes les requêtes d'interruption des unités périphériques indirectes ; or, le PIM ne détecte que le passage de la ligne de niveau "vide" au niveau "occupé". Il y a donc interruption par le PIM quand le nombre des requêtes passe de 0 à 1, mais pas lorsqu'il passe de 1 à 2, de 2 à 3, etc...

1.1.3.4. Empêchement implicite d'interruption

Dans le paragraphe 1.1.3.2., on a vu que le fonctionnement du PIM (l'étape du balayage sur le registre d'interruption pour reconnaître des requêtes) peut être empêché par une instruction. Le fonctionnement du PIM est aussi empêché après l'occurence d'une interruption du processus courant qui entraîne l'exécution d'une instruction "saut avec retour" qui se trouve à une adresse fixe. Cette instruction pointe vers la partie de traitement d'interruption correspondante. Le balayage sur le registre d' interruption ne recommence qu'après l'exécution d'une instruction qui autorise le fonctionnement du PIM.

Ce balayage est également empêché pendant les intervalles de temps suivants :

- {<instruction quelle qu'elle soit>}
- {<instruction d'entrée-sortie> [<instruction d'entrée-sortie>]*
  <une ou deux instructions de type non entrée-sortie>}

Dans ce cas, l'arrêt du fonctionnement du PIM est provisoire et il ne sera pas nécessaire d'exécuter l'instruction qui autorise le balayage du PIM.

1.2. Principes généraux de la gestion des opérations d'entrée-sortie -

Les requêtes d'interruption en provenance d'une unité périphérique peuvent être de natures différentes selon les caractéristiques du fonctionnement de

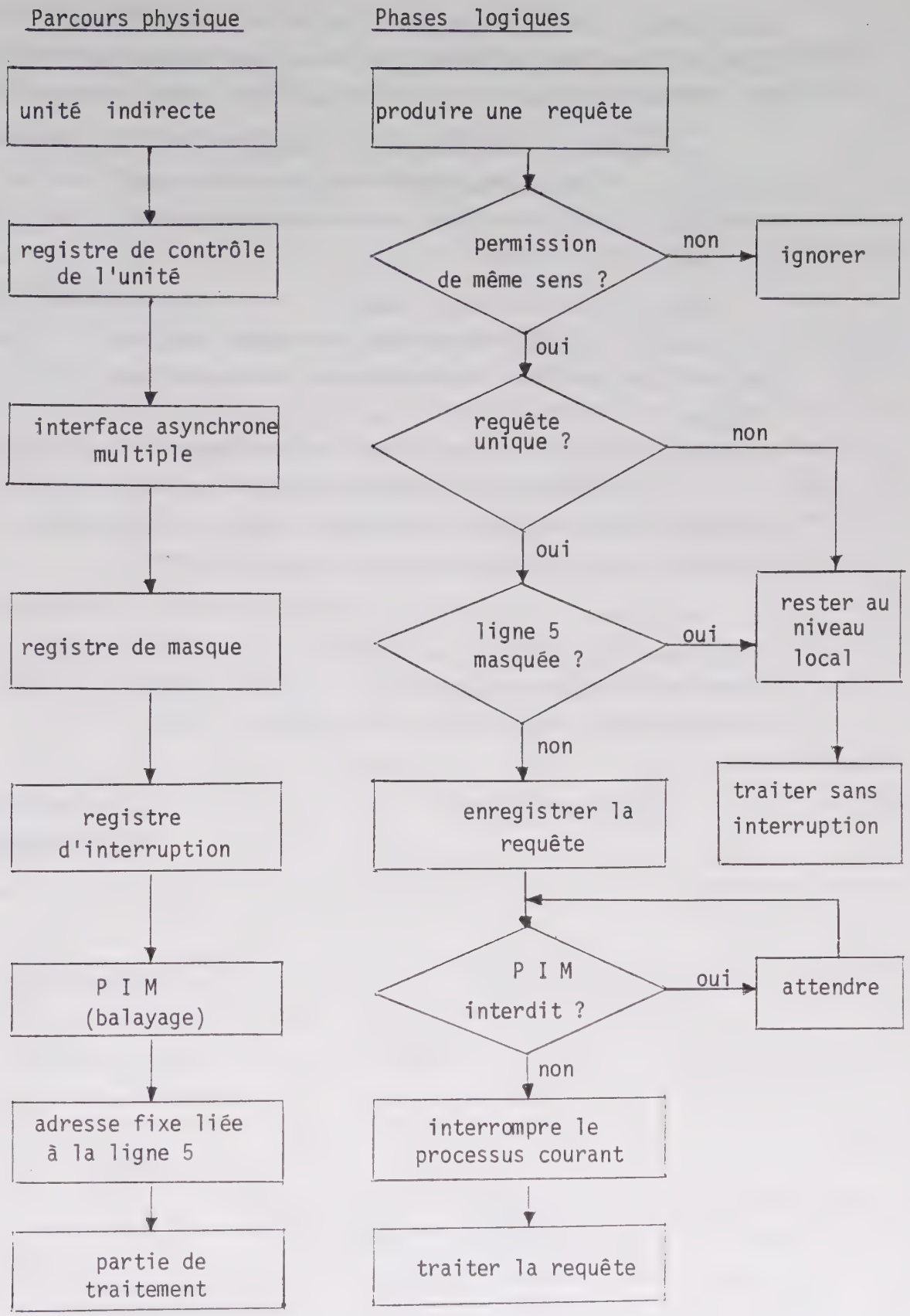


Figure 3.3. - L'apparition et l'empêchement d'interruption d'une unité périphérique indirecte

cette unité : soit l'entrée, soit la sortie d'une unité d'information élémentaire (un caractère ou un mot), soit la fin d'un pas élémentaire de mouvement de la tête de lecture ou d'écriture, soit une anomalie du matériel.

Les requêtes d'interruption se produisent avec un rythme rapide. Tôt ou tard, elles provoqueront des interruptions du processus courant. Cependant, on verra plus loin que certaines de ces requêtes pourront être traitées sans provoquer aucune interruption.

Malgré la diversité des opérations d'entrée-sortie des unités périphériques différentes, la gestion de celles-ci respecte quelques principes communs. La gestion des opérations d'entrée-sortie généralement est réalisée au travers de 3 phases :

- phase 1 : établir les paramètres indispensables à l'exécution d'une opération d'entrée-sortie et déclencher cette opération ;
- phase 2 : traiter les interruptions dues à cette opération ;
- phase 3 : vérifier la fin totale de cette opération.

Les phases 1 et 3 sont à la charge du processus appelant ; quant à la phase 2, elle se réalise à l'insu des processus en utilisant les valeurs des paramètres fournies par la phase 1. La phase 3 examine la valeur d'un paramètre spécifique pour vérifier la fin d'opération (figure 3.4.).

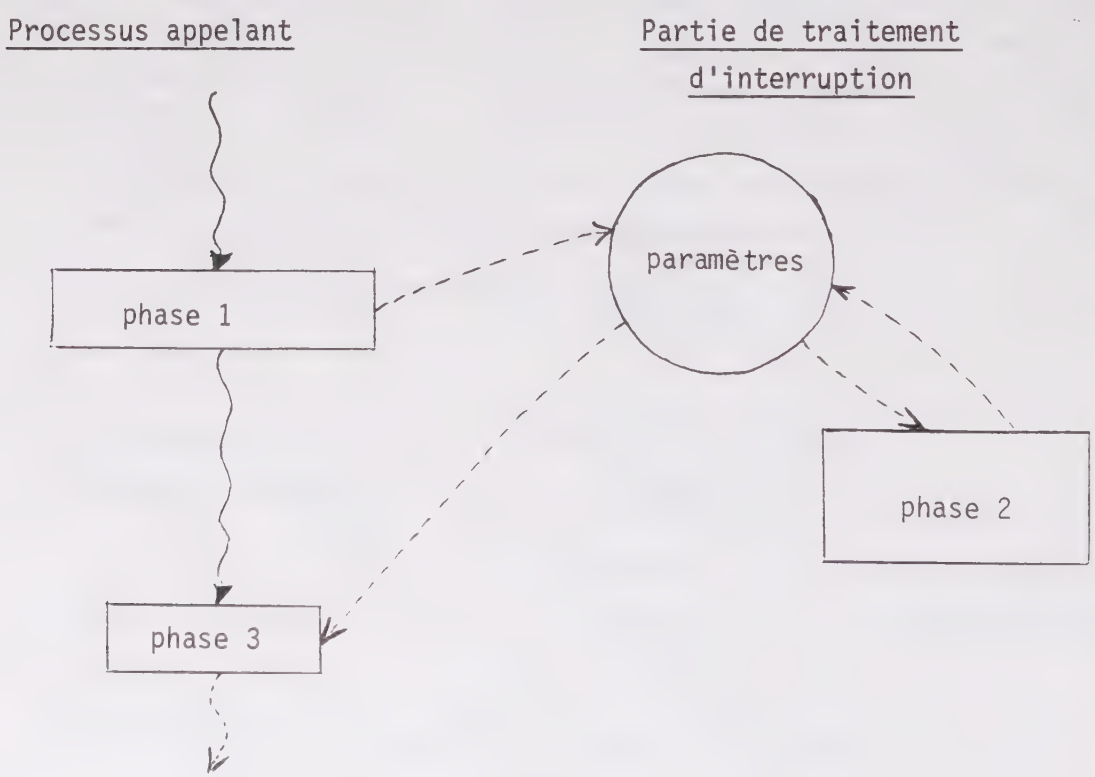


Fig 3.4. - 3 phases d'une opération d'entrée-sortie

1.2.1. Phase 1

La phase 1 correspond à l'exécution d'un sous-programme d'entrée-sortie appelé par le processus en cours. Chaque sous-programme répond à une opération d'entrée-sortie spécifique à une unité périphérique.

A chaque unité périphérique est attachée une variable "état" dont la valeur nulle ou non nulle permet de savoir si celle-ci est libre ou occupée. Chaque valeur non nulle donne le type de l'opération en cours sur cette unité périphérique.

La phase 1 teste tout d'abord la disponibilité de l'unité périphérique concernée, aussi bien au point de vue logique (la valeur de la variable "état") qu'au point de vue physique (l'état du matériel). Si l'unité périphérique n'est pas prête à faire l'opération demandée par le processus en cours, ce dernier se met en attente en donnant le contrôle à un autre processus via "l"ordonnateur" (se référer au chapitre "Gestion des processus concurrents").

Si l'unité périphérique est prête, la phase 1 réalise les pas suivants :

. analyser et sauvegarder les valeurs effectives des paramètres dans une zone à laquelle la phase 2 peut accéder. Parmi ces paramètres, il faut avoir l'adresse d'un mot appelé "signal de fin" choisi par chaque appel de sous-programme d'entrée-sortie. Le contenu de ce mot est mis à 0 impérativement par la phase 1. (Lorsque ce mot aura une valeur non nulle, alors l'opération sera terminée) ;

. mettre une valeur non nulle compatible avec le type de l'opération demandée dans la variable "état" ;

. établir un pointeur vers la partie de traitement d'interrruption correspondant dans une adresse fixe concernant la ligne d'interruption ;

. déclencher enfin l'opération d'entrée-sortie (voir la figure 3.5.).

1.2.2. Phase 2

La phase 2 correspond au traitement de toutes les interruptions liées à une même opération d'entrée-sortie. Chaque partie de ce traitement répond à un opération spécifique parmi les opérations sur une ligne d'interruption.

Une partie de traitement sera invoquée si les deux conditions suivantes sont satisfaites :

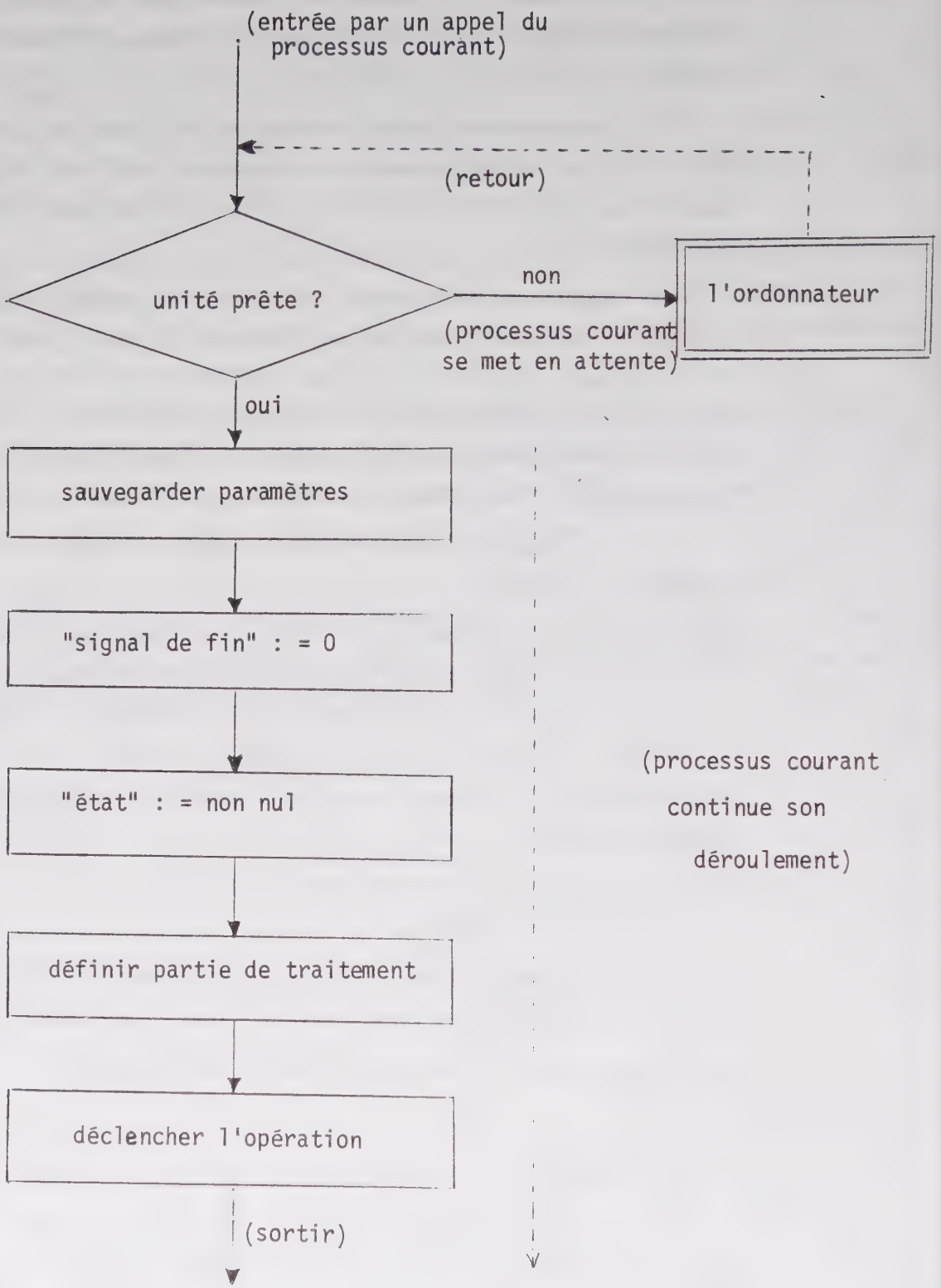


Figure 3.5. - Phase 1 : sous-programme d'entrée-sortie

= une requête d'interruption est reconnue par le balayage du PIM ;

= dans l'adresse fixe liée à la ligne d'interruption en question existe une instruction "saut avec retour" qui pointe vers cette partie.

Pour des considérations de sécurité du système vis-à-vis des interruptions parasitaires provoquées par une anomalie des unités phériphériques, le pointeur vers la partie spécifique de traitement est mis en place seulement au moment où une opération d'entrée-sortie est déclenchée. Après la fin totale de cette opération, ce pointeur est remplacé par un autre pointeur qui pointe vers une séquence neutre. Celle-ci laisse passer les interruptions parasitaires sans les traiter. Dans ce cas, on dit que la ligne d'interruption est invalidée.

Une partie de traitement d'interruption peut également détecter des interruptions parasitaires grâce à la valeur de la variable "état". Une valeur nulle de "état" indique qu'il n'y a aucune opération sur cette unité, c'est-à-dire que l'interruption dans ce cas est bien parasitaire. On la laisse passer sans la traiter.

De plus, une partie de traitement examine les conditions matérielles de l'unité périphérique concernée pour savoir la raison de l'interruption, afin de la traiter convenablement. S'il n'y a pas de concordance entre le type d'opération et la raison d'interruption, l'interruption en question est considérée comme une interruption parasitaire.

La phase 2 réalise avant tout la sauvegarde de l'état actuel du calculateur (le compteur ordinal, des registres) (figure 3.6.). Ensuite, elle reçoit la donnée entrée (dans le cas d'une opération d'entrée), marque l'évolution de l'opération en cours par la modification de la valeur des paramètres, et vérifie la condition de fin totale de cette opération.

Si la fin totale n'est pas encore arrivée, la phase 2 relancera l'opération (dans le cas d'une opération de sortie). Lorsque la fin totale arrive, la phase 2 met une valeur non nulle dans le mot "signal de fin" fourni par la phase 1 et puis libère l'unité périphérique concernée par la remise à zéro de la variable "état" correspondante. En cas d'erreur, la phase 2 essaiera de restaurer les paramètres et de recommencer l'opération d'entrée-sortie en question.

1.2.3. Phase 3

Cette phase a pour but de s'assurer qu'une opération d'entrée-sortie déclenchée a été réellement terminée. Elle n'a aucune influence sur la réalisation des opérations d'entrée-sortie. Elle est indispensable entre une lecture des informations à partir des unités périphériques et le traitement de ces informations en mémoire.

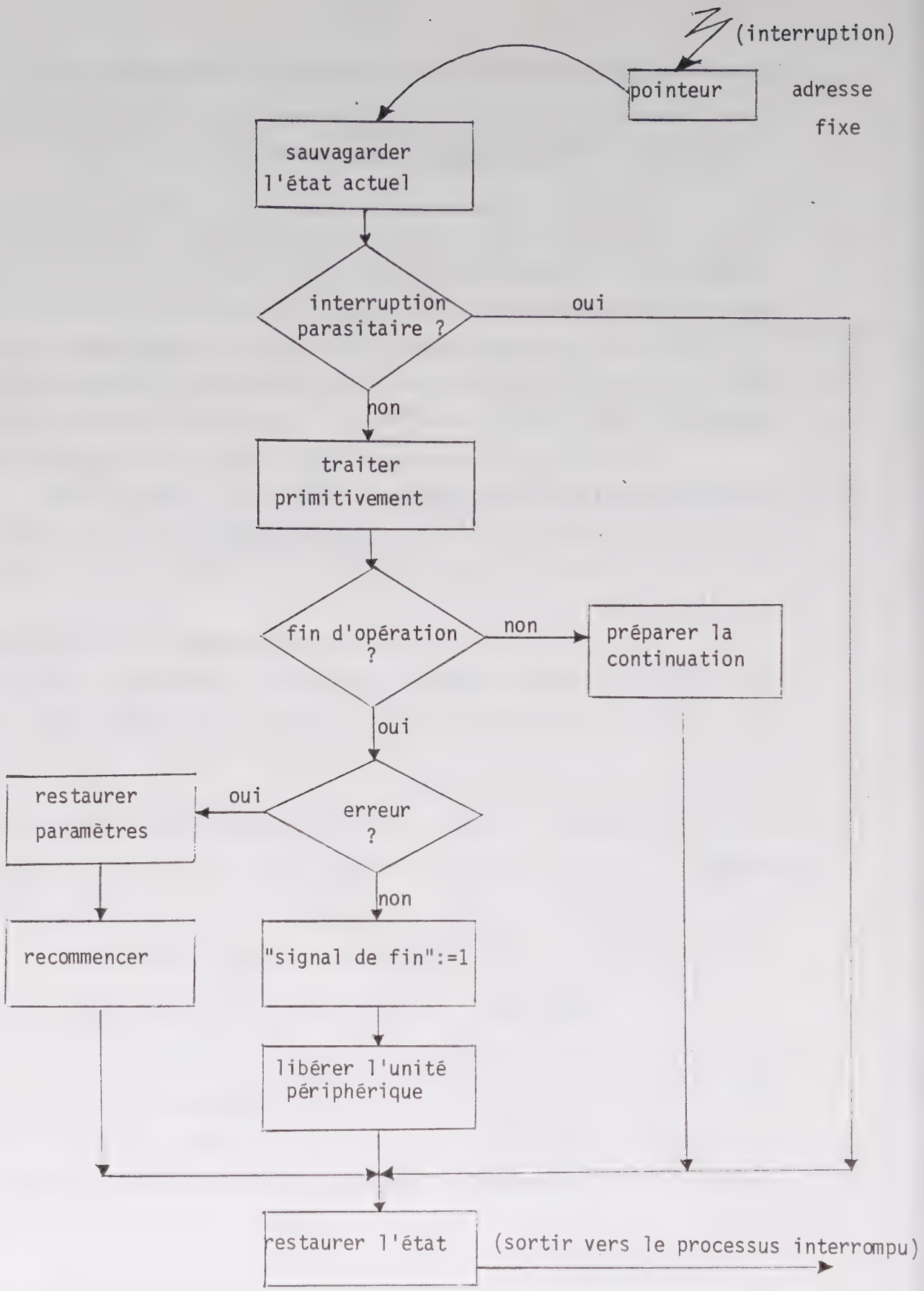


Figure 3.6. - Phase 2 : traitement d'une interruption

La phase 3 est l'exécution d'un sous-programme d'attente accessible par un appel du processus en cours d'exécution. C'est un module unique pour tous les types des opérations d'entrée-sortie et pour toutes les unités périphériques. Celui-ci a un seul paramètre : l'adresse du mot "signal de fin" choisi par la phase 1 de l'opération d'entrée-sortie que l'on veut vérifier.

Le sous-programme d'attente est un module commun, il peut être appelé simultanément par des processus concurrents. Cependant, il est impossible d'organiser des sous-programmes réentrants avec le jeu d'instructions du langage d'assemblage du calculateur VARIAN. Ce sous-programme d'attente devient donc une section critique, et il faut s'assurer qu'aucun autre appel à ce sous-programme n'a lieu pendant qu'un processus se trouve encore à l'intérieur de celui-ci. Par conséquent, il est interdit de changer de processus à l'intérieur de ce sous-programme.

D'autre part, pour la raison de la simultanéité de fonctionnement des processus, si la fin d'une opération d'entrée-sortie attendue par un processus n'est pas encore arrivée, le processus en question est obligé de se mettre en attente en donnant le contrôle à un autre processus via l'ordonnateur. Le changement de processus doit donc être réalisé par ce sous-programme, mais en dehors de ce dernier.

C'est pourquoi l'exécution du sous-programme d'attente est la suivante :

le processus en cours appelle et exécute ce sous-programme. La valeur de la variable "signal de fin" choisie pour paramètre est examinée. Si cette valeur est non nulle, le processus en cours sort de ce sous-programme et continue son déroulement. Au contraire, si cette valeur est encore nulle :

- d'une part, l'adresse de retour après l'exécution de ce sous-programme, normalement vers le processus appelant, est remplacée par l'adresse de l'ordonnateur,
- et, d'autre part, l'adresse de l'instruction d'appel à ce sous-programme dans le processus appelant est sauvegardée comme le compteur ordinal actuel du processus en cours;

enfin, le processus en cours sort de ce sous-programme en donnant le contrôle à l'ordonnateur.

L'ordonnateur sauvegardera l'état de ce processus avec le compteur ordinal modifié pour le restaurer ultérieurement (voir le fonctionnement de l'ordonnateur dans le chapitre"Gestion des processus concurrents").

Aussi, lorsque ce processus reprendra le contrôle, il rappellera de nouveau ce sous-programme d'attente (voir la figure 3.7.).

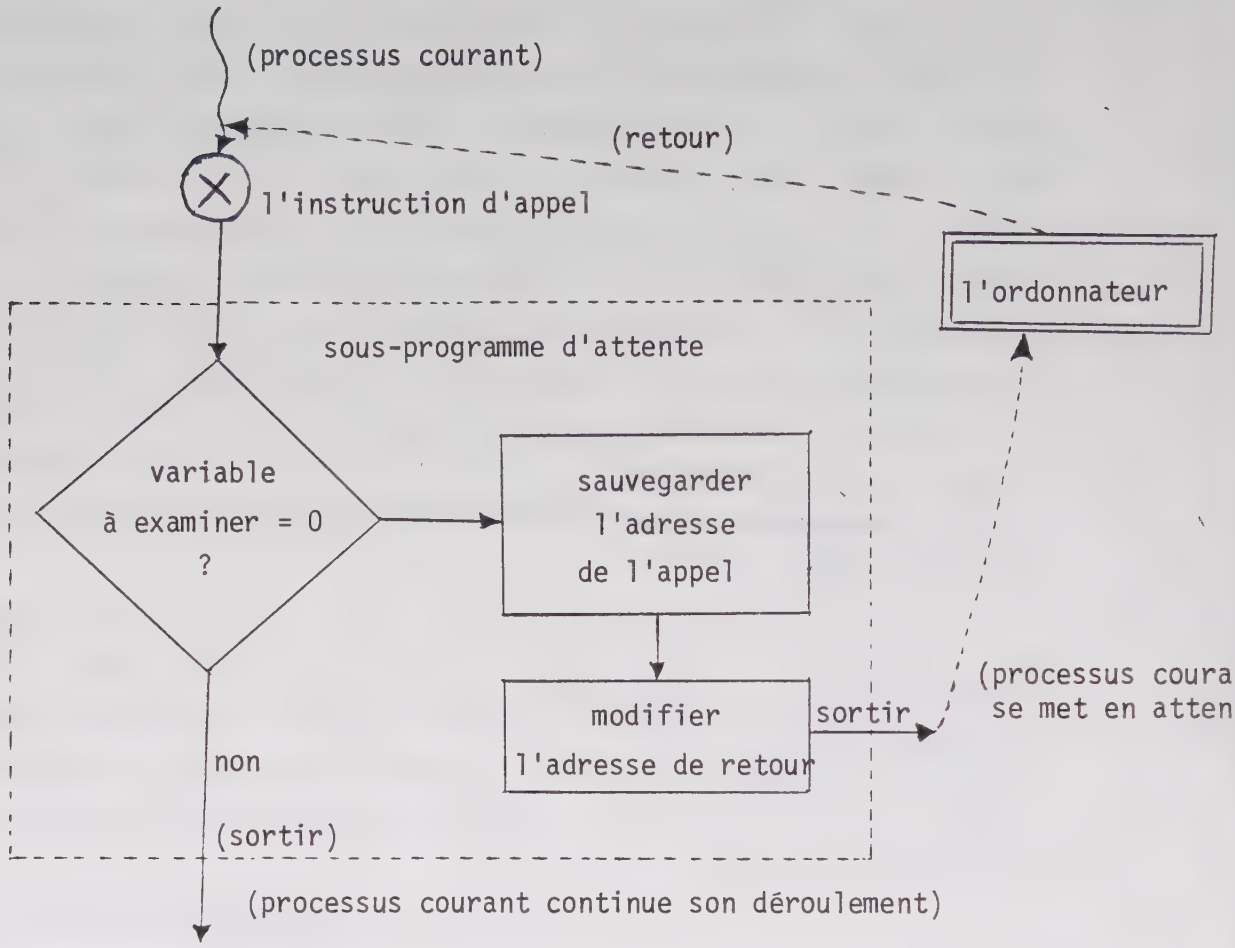


Figure 3.7. Phase 3 : sous-programme d' attente (dans le déroulement d'un processus).

1.3. Relations mutuelles

1.3.1. Relation entre un processus et ses opérations d'entrée-sortie

Le processus en cours déclenche une opération d'entrée-sortie sur une unité périphérique en activant la phase 1. Après quoi, cette unité périphérique fonctionne de façon autonome, indépendante et parallèle avec le processus courant, grâce au traitement des interruptions réalisé au cours de la phase 2. Le processus en cours peut déclencher consécutivement plusieurs opérations d'entrée-sortie différentes, et même s'il perd le contrôle, toutes les opération d'entrée-sortie qui ont été déclenchées par lui se poursuivront jusqu'à la fin totale de chacune d'elles.

Un processus peut perdre le contrôle au cours de la phase 1 et de la phase 3, soit dans le cas de l'attente de la disponibilité d'une unité périphérique, soit dans le cas de l'attente de la fin totale d'une opération d'entrée-sortie.

1.3.2. Relation entre les opérations d'entrée-sortie

Les unités périphériques peuvent fonctionner indépendamment entre elles. Comme on l'a vu au paragraphe 1.1.3.4., le traitement d'une interruption entraîne la suspension de fonctionnement du PIM. Ceci cause souvent un retard inacceptable de traitement des autres requêtes d'interruption.

Pour résoudre ce problème, pendant l'exécution de la partie de traitement d'interruption de certaines lignes, d'une part, on ignore les interruptions de cette ligne (par la mise à la valeur 1 du bit concernant cette ligne dans le registre de masque), d'autre part, on autorise le fonctionnement du PIM (figure 3.8.).

En conséquence, le traitement d'interruption d'une ligne peut être interrompu par des requêtes d'interruption des autres lignes, quels que soient leurs niveaux de priorité.

Par exemple, on examine le traitement d'une ligne d'interruption numéro i_1 quelconque parmi n lignes en totalité.

$$i_1 \in \{\text{ensemble des lignes : } 1, 2, \ldots, n\}$$

Ce traitement pourra être interrompu par une requête de la ligne i_2.

$$i_2 \in [\{\text{ensemble des lignes}\} - i_1]$$

A son tour, le traitement de la ligne i_2 pourra être interrompu par une requête d'interruption de la ligne i_3.

$$i_3 \in [\{\text{ensemble des lignes}\} - (i_1, i_2)]$$

Les interruptions peuvent continuer à se produire de cette manière jusqu'au moment où :

$$[\{\text{ensemble des lignes}\} - (i_1, i_2, \ldots, i_n)] = \emptyset \text{ (vide)}$$

(Voir la figure 3.9.).

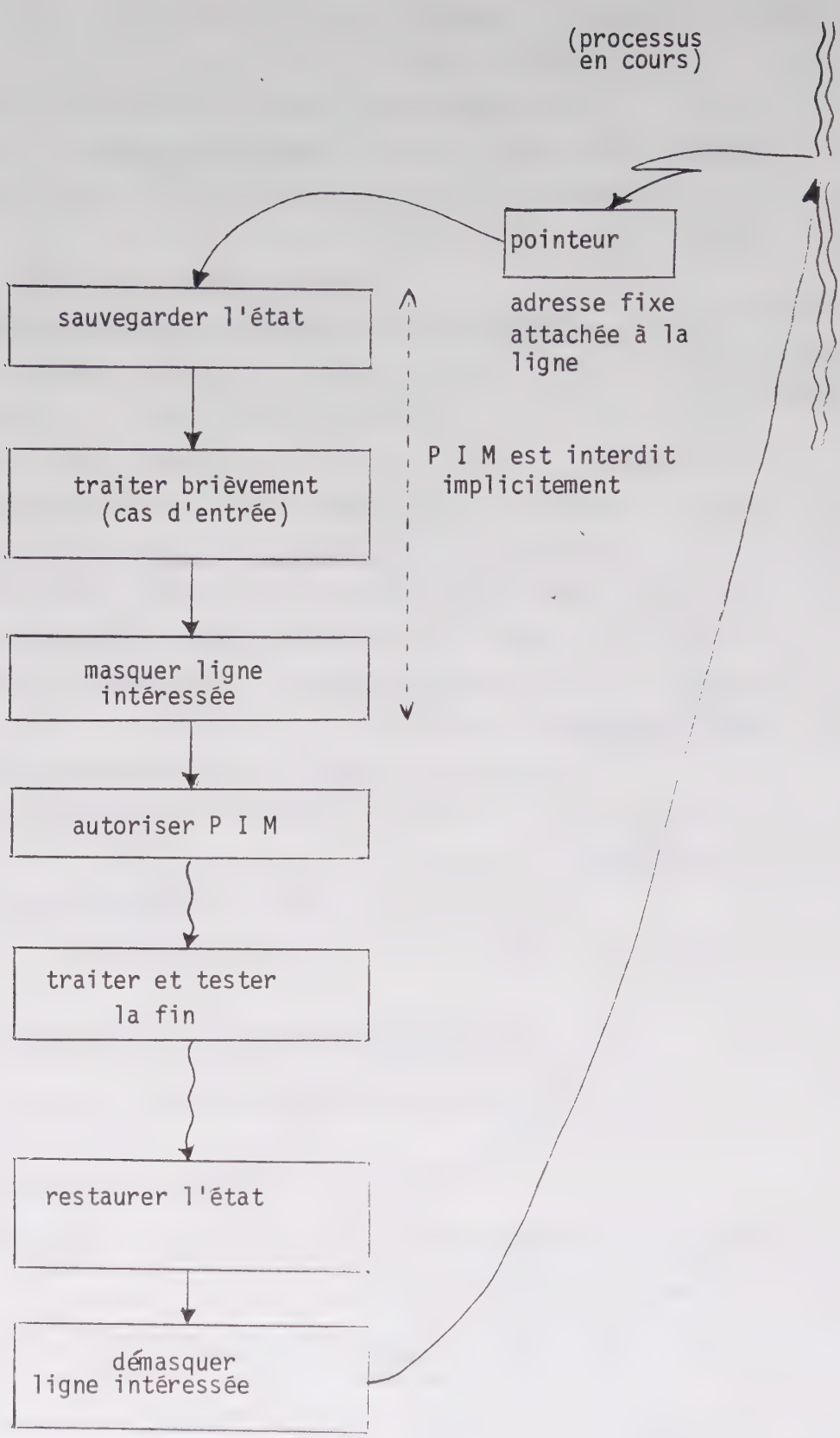


Figure 3.8. - Traitement d'une ligne d'interruption

Il est à noter que, grâce à cette méthode, le traitement d'interruption d'une ligne ne pourra jamais être interrompu par des requêtes d'interruption de la ligne en question.

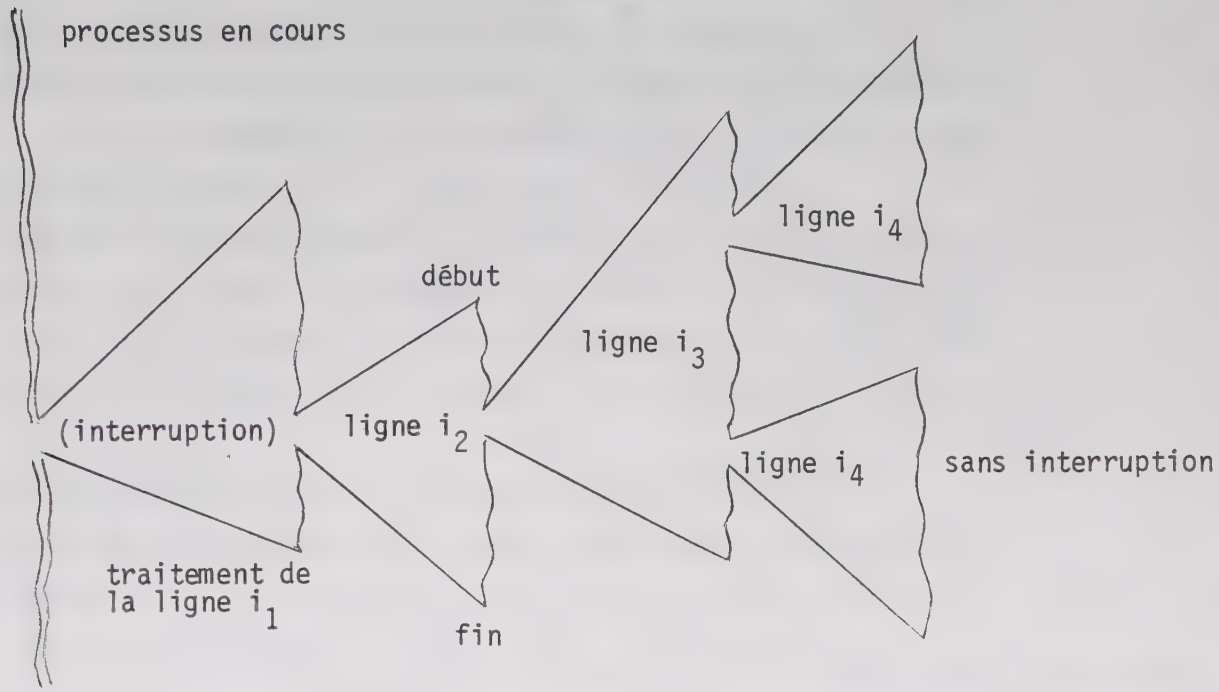


Fig 3.9. Relation entre les traitements des lignes d'interruption (avec hypothèse n = 4)

1.3.3. Relation entre l'horloge et les unités périphériques

On a vu que le fonctionnement du PIM est interdit implicitement dès le début de traitement d'une interruption, y compris le traitement d'une interruption de l'horloge. Il faut donc impérativement autoriser à nouveau le fonctionnement du PIM quelque part à l'intérieur de chaque partie de traitement d'interruption. De plus, cette autorisation doit avoir lieu le plus tôt possible pour éviter l'écrasement des requêtes d'interruption en attente par l'arrivée des nouvelles requêtes de même ligne.

Au contraire, une interruption de l'horloge n'est jamais interdite implicitement. Alors, dès le début de chaque partie de traitement d'interruption, il faut empêcher explicitement l'interruption de l'horloge (RTC), afin que la sauvegarde de l'état actuel s'accomplisse entièrement. Par conséquent,

avant de sortir de la partie de traitement, on remettra en place la possibilité d'avoir l'interruption de l'horloge.

La partie de traitement d'interruption de l'horloge et les parties de traitement d'interruption des unités périphériques respectent le principe décrit par la figure 3.10.(a).

Cependant, il est à noter que l'on peut avoir le cas d'une interruption de l'horloge pendant le traitement d'une interruption d'une unité périphérique. La figure 3.10.(b) nous montre ce phénomène.

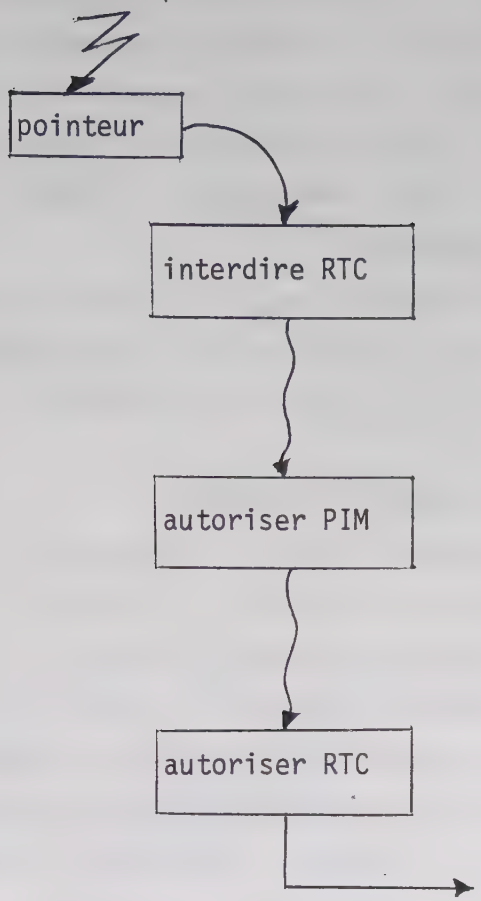
Après que PIM soit autorisé à fonctionner, une requête d'interruption d'une ligne i_2 peut provoquer une interruption qui interrompt le traitement de la ligne i_1. A la fin du traitement de la ligne i_2, RTC est autorisé. Après quoi, pendant le reste du traitement d'interruption de la ligne i_1, une interruption de l'horloge peut donc se produire, ceci avant l'exécution de l'instruction qui l'autorise.

En conclusion, une interruption de l'horloge peut se produire pendant le traitement d'interruption d'une unité périphérique. De même, une interruption d'une unité périphérique peut se produire pendant le traitement d'interruption de l'horloge. Cependant, il est important d'empêcher une interruption de l'horloge pendant le traitement d'une interruption de l'horloge, parce que chaque partie de traitement d'interruption est invoquée comme un sous-programme et les sous-programmes ne sont pas réentrants.

Cet empêchement ne peut être assuré uniquement par l'instruction qui interdit RTC ou par le registre de masque dans la partie de traitement d'interruption de l'horloge.

La solution donnée dans le paragraphe précédent (voir la figure 3.9.) nous montre que l'on a une chaîne finie avec un nombre de maillons inférieur ou égal au nombre total des lignes d'interruption. Par conséquent, le traitement d'interruption de l'horloge se terminera dans un temps fini.

Pour éviter qu'une interruption de l'horloge se produise pendant un traitement d'une autre interruption de l'horloge, il suffit d'avoir un intervalle de temps assez grand entre deux interruptions consécutives de l'horloge, i.e., pour deux débordements consécutifs du compteur externe du RTC. Pour cela, on choisit une valeur initiale convenable pour le compteur externe du RTC, et cette valeur est remise chaque fois qu'une interruption de l'horloge se produit. A présent, l'intervalle de temps choisi est environ une demi seconde.



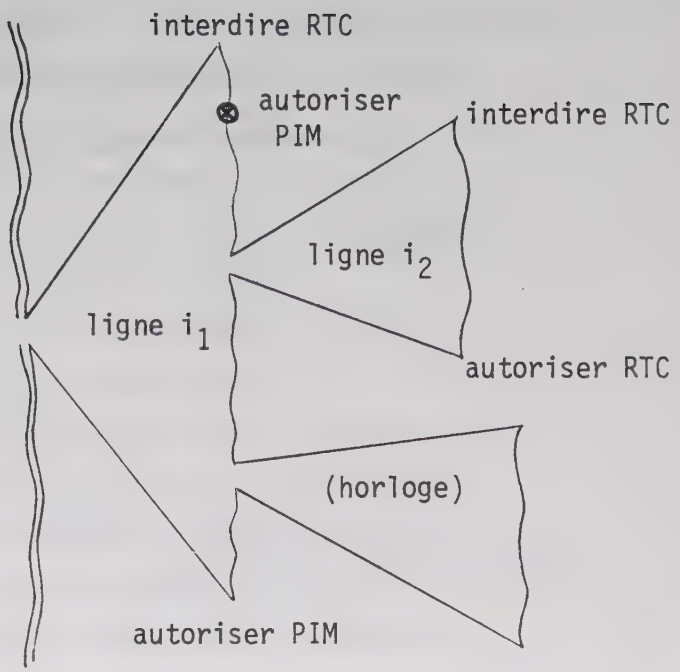


Figure 3.10(a) Partie de traitement d'une interruption quelle qu'elle soit.

(b) L'interruption de l'horloge se produit avant l'autorisation dans la partie de traitement.

2 - TRAITEMENT DES LIGNES D'INTERRUPTION

Dans les paragraphes qui suivent, on va souligner les caractéristiques de traitement de chaque ligne d'interruption. Les lignes sont numérotées par le numéro du bit correspondant dans le registre de masque.

2.1. Ligne 1 : Microdensitomètre

Cette ligne est dédiée au microdensitomètre ; alors, elle appartient à un seul processus. Elle sert à deux types d'opération du microdensitomètre : le mouvement horizontal et la lecture verticale des images photographiques montées sur un tambour creux tournant en permanence (voir l'article "Le microdensitomètre" de D. TAUPIN, C. ZELWER, J. NOGUES, A. GUINIER, Le Courrier du CNRS - janvier 1975).

La phase 1 de mouvement indique la position finale à atteindre. En ce qui concerne la lecture, l'unité d'information d'échange est un mot et le microdensitomètre a un tampon d'entrée dont la taille est un mot de 16 bits, soit 2 octets de mesure. La phase 1 de la lecture indique sur la zone lisible d'une ligne verticale le nombre de mots à sauter (sans lire), le nombre de mots à lire et la zone de mémoire recevant des données.

Chaque phase 1 pour chaque opération met en place l'adresse de la partie de traitement qui correspond à cette opération. Pour le microdensitomètre, il y a donc 2 parties différentes de traitement d'interruption.

2.2.1. Partie de traitement d'interruption de mouvement

Le mouvement de la tête de lecture à partir de la position actuelle (au moment du déclenchement du mouvement dans la phase 1) à la position finale (indiquée par la phase 1) se réalise par des pas élémentaires dont la longueur est fixe, sauf pour le dernier pas qui peut être plus petit (figure 3.11.(a)).

Une requête d'interruption se produit sur la ligne 1 au moment où la tête de lecture finit un pas élémentaire. Le traitement de cette requête aura pour but de relancer, si nécessaire, le mouvement de la tête de lecture en calculant la longueur et le sens du pas suivant (figure 3.11.(b)).

2.1.2. Partie de traitement d'interruption de lecture

La lecture du microdensitomètre se réalise au moyen d'une tête de lecture translatable et d'un tambour creux qui tourne sans cesse autour d'un axe parallèle à la direction de translation de la tête de lecture. Chaque cercle du tambour est divisé en une zone illisible et une zone lisible. Dans la zone lisible, on met un film à analyser de façon optique. Quand la lecture est déclenchée par la phase 1, la lecture ne commence vraiment qu'après le début de la zone lisible (figure 3.12.(a)).

Chaque fois que la tête de lecture enregistre deux octets d'information dans la zone lisible, ces données sont stockées dans le tampon du microdensitomètre ; ce tampon devient "plein" et une requête d'interruption se produit. La phase 2 doit savoir distinguer les données à sauter et les données à lire.

Parce que le rythme d'arrivée des données est très rapide (le tambou tourne à une vitesse comprise entre 1/4 et 1 tour par seconde), 2 phénomènes son à envisager :

- une nouvelle donnée arrive quand une ancienne est encore dans le tampon, et écrasera l'ancienne. C'est un phénomène catastrophique (condition "overrun"), la phase 2 doit donc pouvoir le détecter et ensuite, recommencer la lecture ;

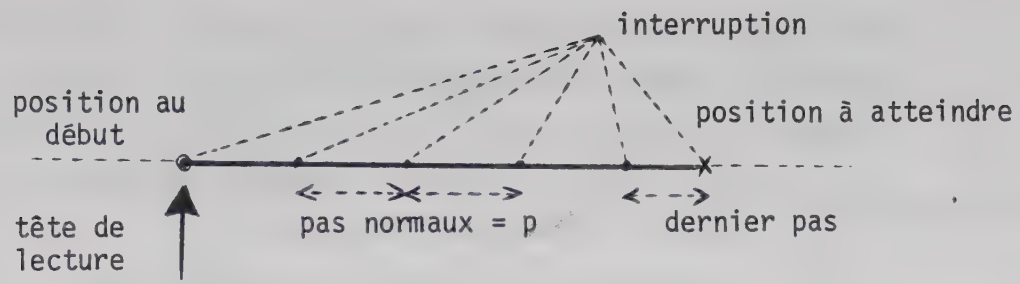


Figure 3.11 (a) Le mouvement de la tête de lecture du microdensitomètre

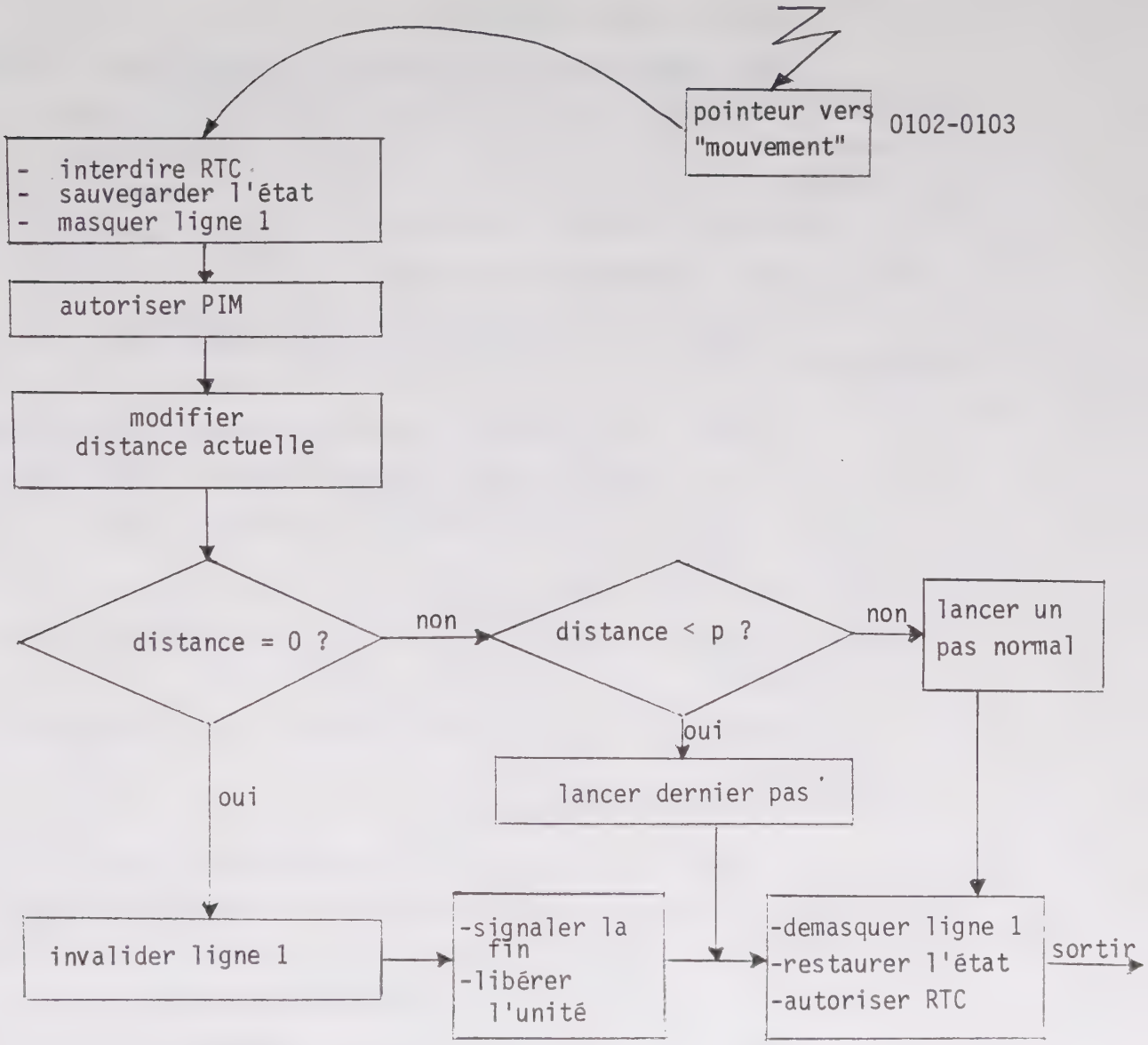


Figure 3.11 (b) Traitement d'interruption du mouvement de la tête de lecture

- une nouvelle donnée arrive quand le tampon est vide mais alors que le traitement de la donnée précédente n'est pas encore terminé, de sorte que la nouvelle requête d'interruption est ignorée. Pour l'éviter, il faut, avant de sortir de la phase 2, tester l'état du tampon pour reconnaître cette nouvelle donnée et la traiter sans besoin d'interruption.

Après avoir lu le nombre nécessaire de mots à lire, la phase 2 inhibe tout de suite la lecture par le microdensitomètre (figure 3.12.(b)). A partir de ce moment-là, il n'y aura plus de requête d'interruption, bien que la tête de lecture puisse se trouver encore dans la zone lisible.

2.2. Ligne 2 : bande magnétique

Cette ligne est dédiée au dérouleur de bande ; aussi appartient-elle à un seul processus. Elle sert à 3 types d'opération de la bande : le mouvement, la lecture d'un enregistrement et l'écriture d'un enregistrement. Pour la lecture et l'écriture, l'unité d'échange entre la bande et la mémoire centrale est le mot. Le dérouleur a un tampon d'entrée et un tampon de sortie ayant chacun une taille de 1 mot et permettant le découpage en octets sur la bande.

La phase 1 de mouvement précise le mouvement à réaliser :rebobiner, sauter un enregistrement en avant ou en arrière, marquer la fin de fichier.

La phase 1 de la lecture d'un enregistrement indique le nombre de mots à lire sur cet enregistrement, la zone de mémoire recevant les données lues et l'adresse d'une variable qui contiendra le code de retour. A la fin de cette opération, le code de retour indique si :

- la lecture s'est terminée normalement,
- la lecture s'est terminée anormalement par une erreur matérielle (erreur de parité),
- la longueur effective de l'enregistrement lu est différente de la longueur demandée,
- la fin de fichier est atteinte.

La phase 1 de l'écriture d'un enregistrement indique la longueur (en unité de mot) de l'enregistrement à écrire, la zone de mémoire contenant les données à écrire et l'adresse d'une variable contenant le code de retour. Le code de retour permet de savoir s'il y a une erreur de parité.

Chaque phase 1 de chaque opération détermine l'adresse de la partie de traitement qui correspond à cette opération. Pour la bande, il y a 3 parties différentes de traitement.

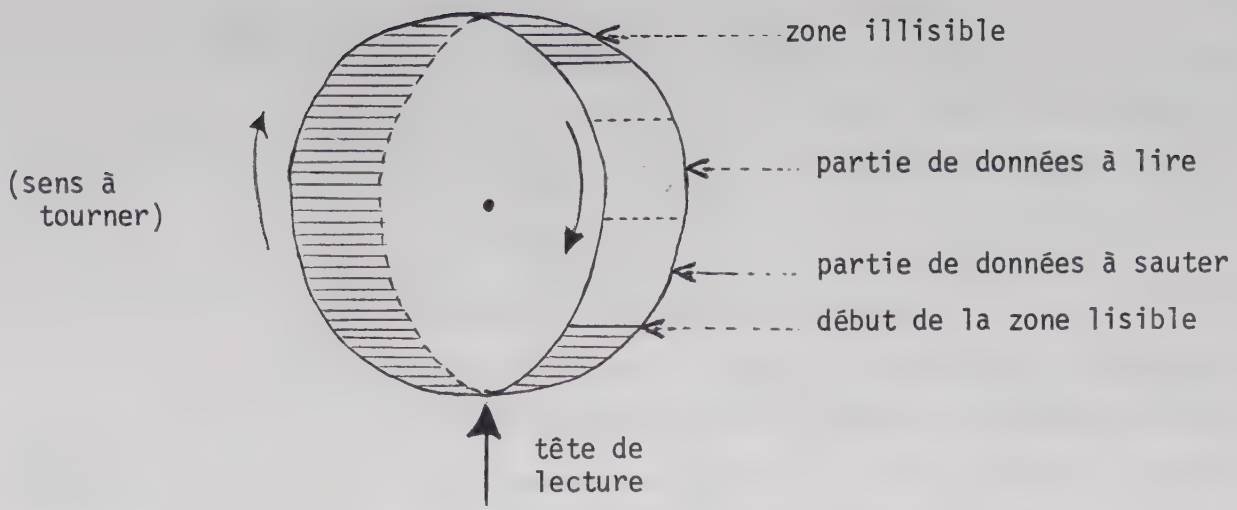
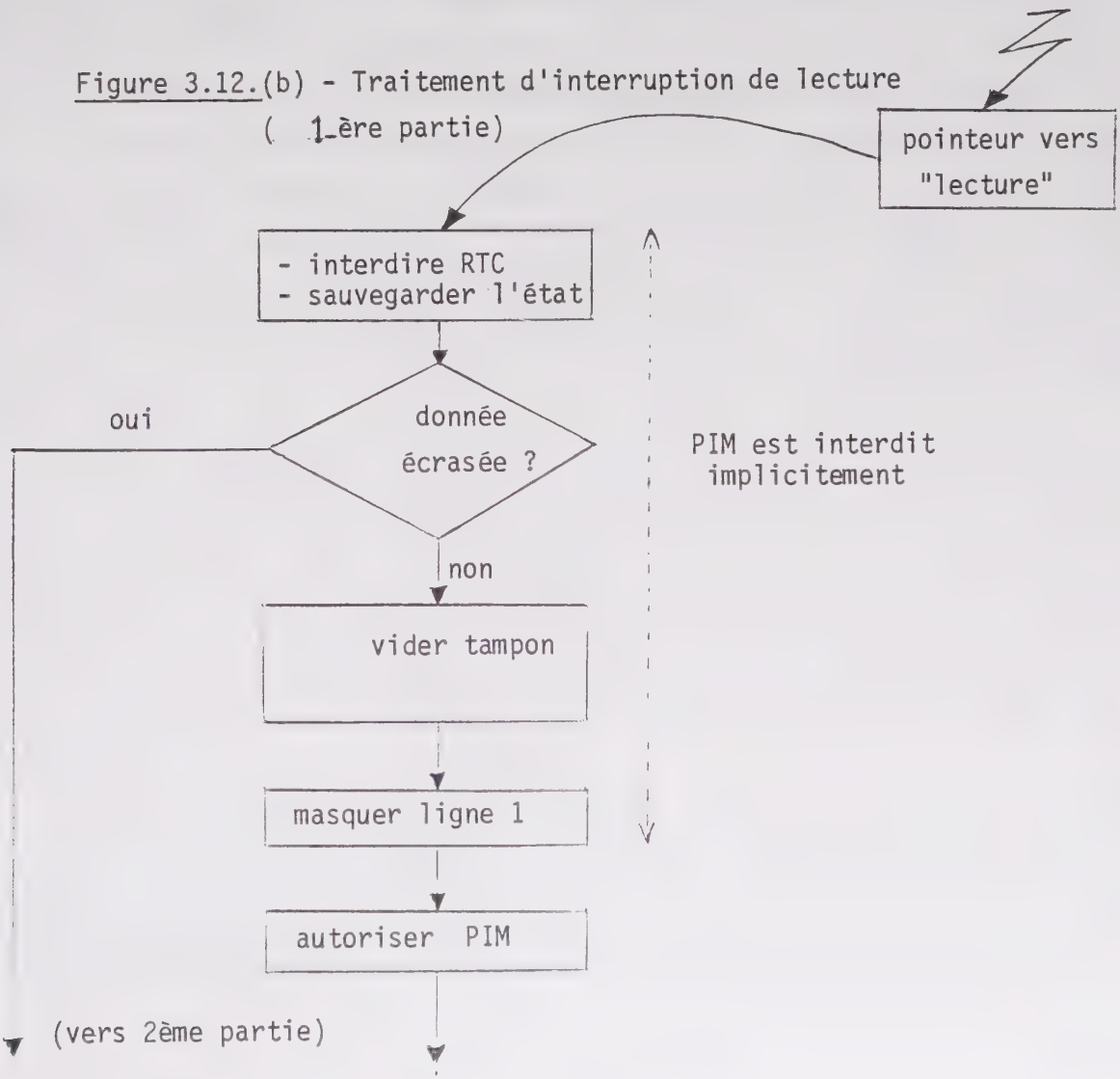


Figure 3.12.(a) - La lecture du microdensitomètre

Figure 3.12.(b) - Traitement d'interruption de lecture
(1ère partie)



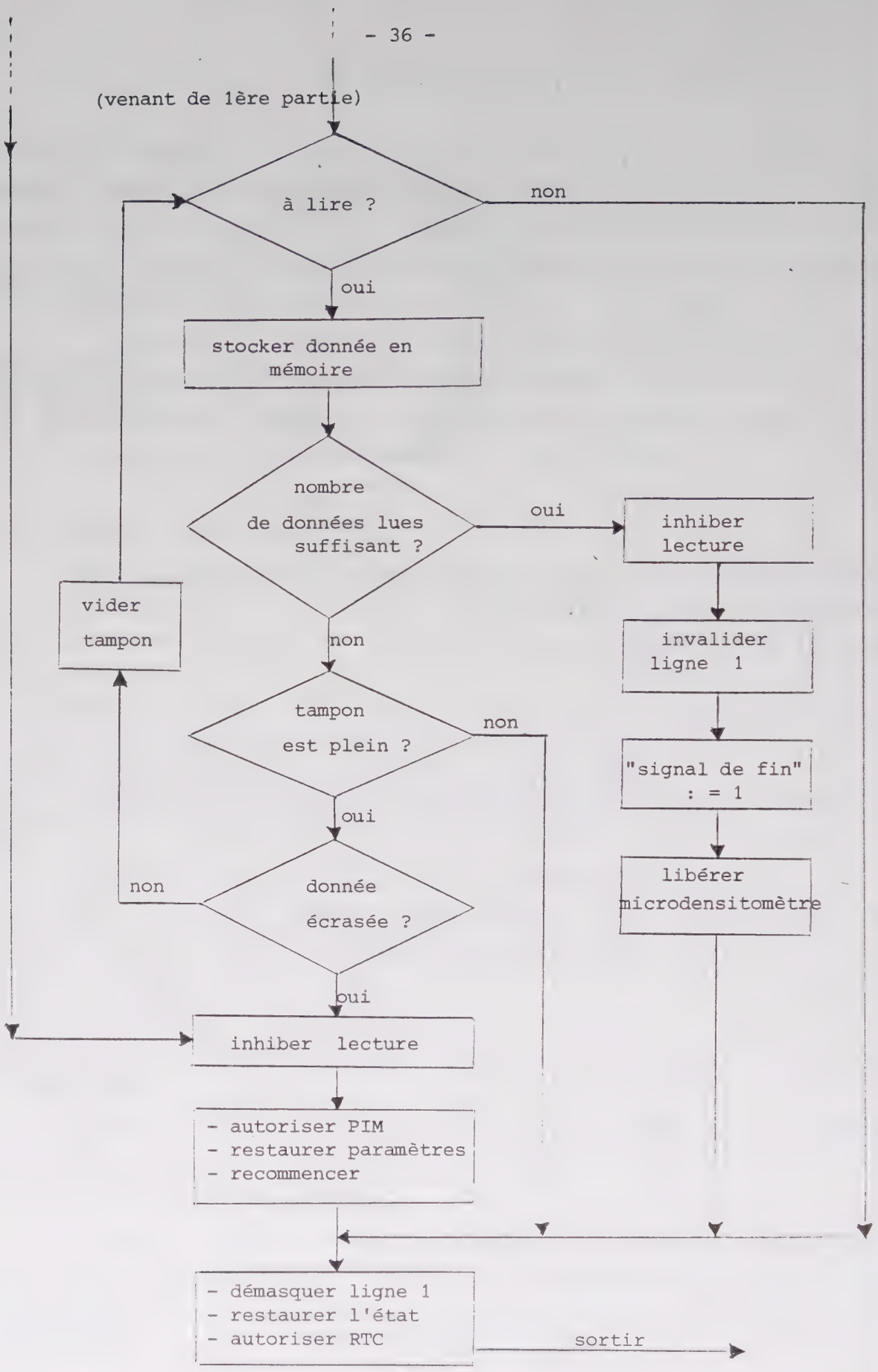


Figure 3.12(b) - Traitement d'interruption de lecture (deuxième partie)

2.2.1. Partie de traitement d'interruption de mouvement

Une seule requête d'interruption se produit au moment où le dérouleur de bande change d'état : de l'état non-prêt à l'état prêt pour une autre opération. Cette requête montre aussi la fin totale de mouvement quel qu'il soit.

2.2.2. Partie de traitement d'interruption de lecture

La lecture de la bande entraîne toujours implicitement un mouvement en avant de la bande. Une requête d'interruption se produit aux moments suivants :

. une unité d'information sur la bande (1 mot) est lue dans le tampon d'entrée du dérouleur, le tampon d'entrée devient donc plein ;

. le mouvement de la bande arrive à la fin d'enregistrement.

Ainsi a-t-on des interruptions de 2 natures différentes pour une même ligne ; il faut donc vérifier l'état du dérouleur pour reconnaître la nature de chaque interruption.

Parce que le rythme d'arrivée des données est très rapide et qu'il n'y a pas de moyen de détecter la perte de donnée au cas où une nouvelle donnée arrive avant que l'ancienne ne soit ôtée du tampon d'entrée, il faut que le traitement d'une interruption soit le plus rapide possible. On interdit donc que les interruptions des autres unités périphériques viennent interrompre le traitement d'une interruption de lecture de la bande. Pour cela, on interdit le fonctionnement du PIM, et, par conséquent, on n'a plus besoin de masquer la ligne d'interruption de bande.

En cas d'erreur de parité de la bande, la phase 2 effectuera tout d'abord un saut en arrière d'un enregistrement, puis une relecture. Si après avoir tenté de refaire la lecture un nombre fini de fois sans succès, la phase 2 affirmera la situation d'erreur de parité de la bande. Evidemment, la récupération d'erreur de parité demande un intervalle de temps considérable, c'est pourquoi la phase 2 réalise le saut en arrière en autorisant le fonctionnement du PIM (figure 3.13.).

Il est rappelé que, pour gagner du temps, la ligne d'interruption de bande n'est pas masquée pendant le traitement d'interruption de cette ligne. Or, pendant la récupération d'erreur de parité, le fonctionnement du PIM est autorisé, le mouvement en arrière de la bande causera donc une interruption.

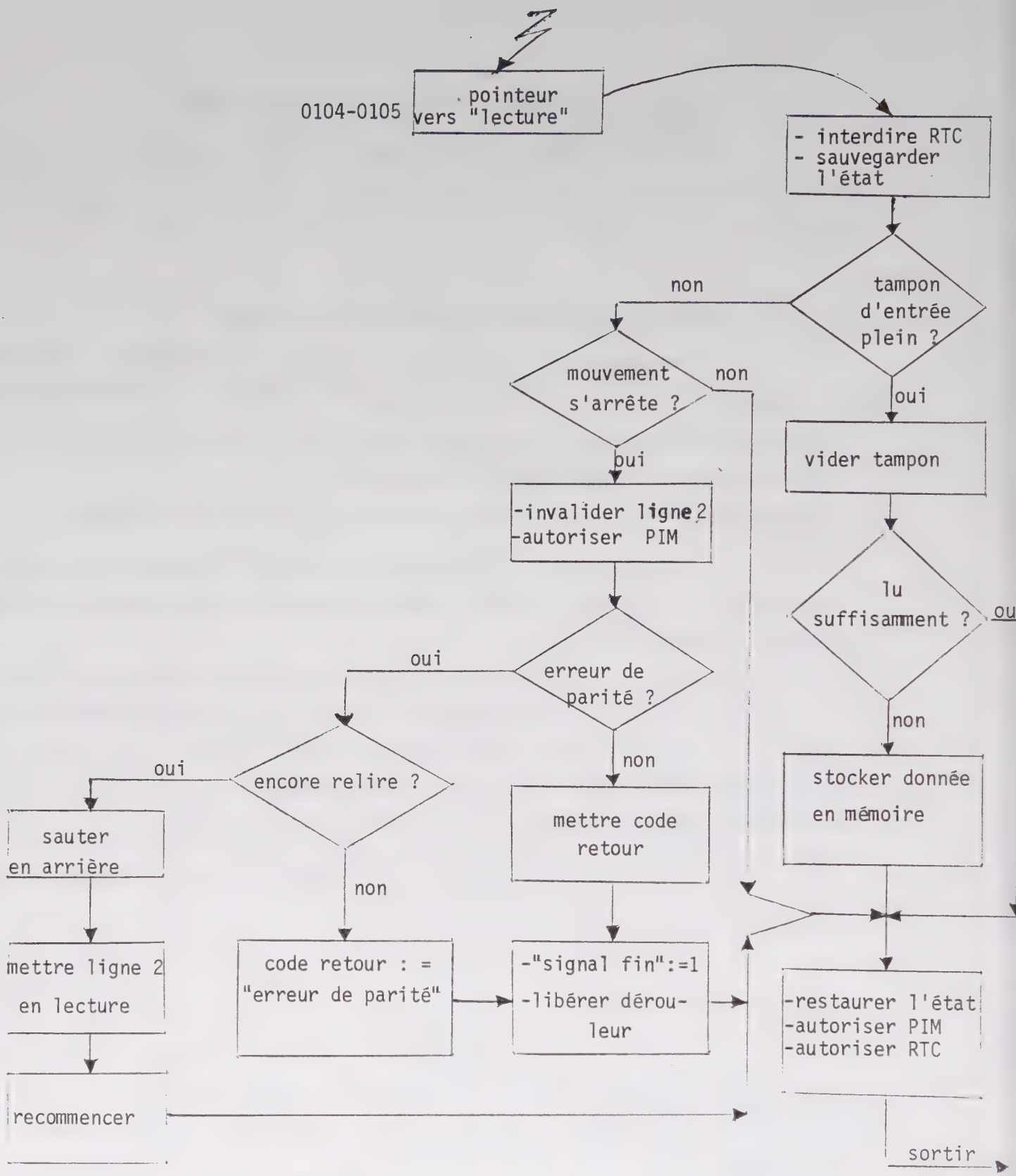


Figure 3.13. - Traitement d'interruption de lecture de la bande

Il est interdit d'invoquer une partie de traitement d'interruption lorsque celle-ci est en cours d'exécution , parce que les parties de traitement d'interruption ne sont pas réentrantes. C'est pourquoi pendant le mouvement en arrière de la bande, il faut invalider la ligne 2 par la mise à l'adresse fixe attachée à la ligne 2 d'un pointeur vers la partie de traitement spéciale qui laisse passer les interruptions sans les traiter. Cette partie spéciale autorise à nouveau le fonctionnement du PIM.

2.2.3. Partie de traitement d'interruption d'écriture

Comme le cas de lecture, l'écriture de la bande entraîne implicitement un mouvement en avant de la bande. Une requête d'interruption se produit aux moments suivants :

- la donnée dans le tampon de sortie du dérouleur est écrite sur la bande ; autrement dit, ce tampon devient vide ;
- le mouvement de la bande s'arrête.

Le mouvement de la bande dépend de la mise des données (à partir de la mémoire) dans le tampon de sortie du dérouleur. Le rythme de cette mise doit être suffisamment rapide pour que le mouvement de la bande ne s'arrête pas avant le dernier mot de l'enregistrement à écrire sur la bande. Autrement dit, après le vidage du tampon de sortie (qui cause une requête d'interruption), il faut mettre dans ce tampon une autre donnée le plus rapidement possible.

Pour cette raison, le principe général de la phase 2 de l'écriture ressemble à celui de la lecture : il n'autorise pas le fonctionnement du PIM pendant le traitement normal, mais, en cas d'erreur de parité, pendant le saut en arrière pour préparer la réécriture, il autorise le fonctionnement du PIM en invalidant les interruptions de la bande.

2.3. Ligne 3 : disque

Cette ligne est dédiée au disque. Elle sert à deux types d'opération : la lecture et l'écriture. La phase 1 d'une opération du disque indique la zone de mémoire et la zone sur le disque.

Le disque est une unité périphérique partageable; la phase 1 peut donc être appelée par plusieurs processus concurrents. Pour éviter qu'un processus attende à l'intérieur de la phase 1 dans le cas où le disque n'est pas disponible, la phase 1 des opérations du disque a le même principe de structure que la phase 3 dans le paragraphe 1.2.3. (figure 3.14).

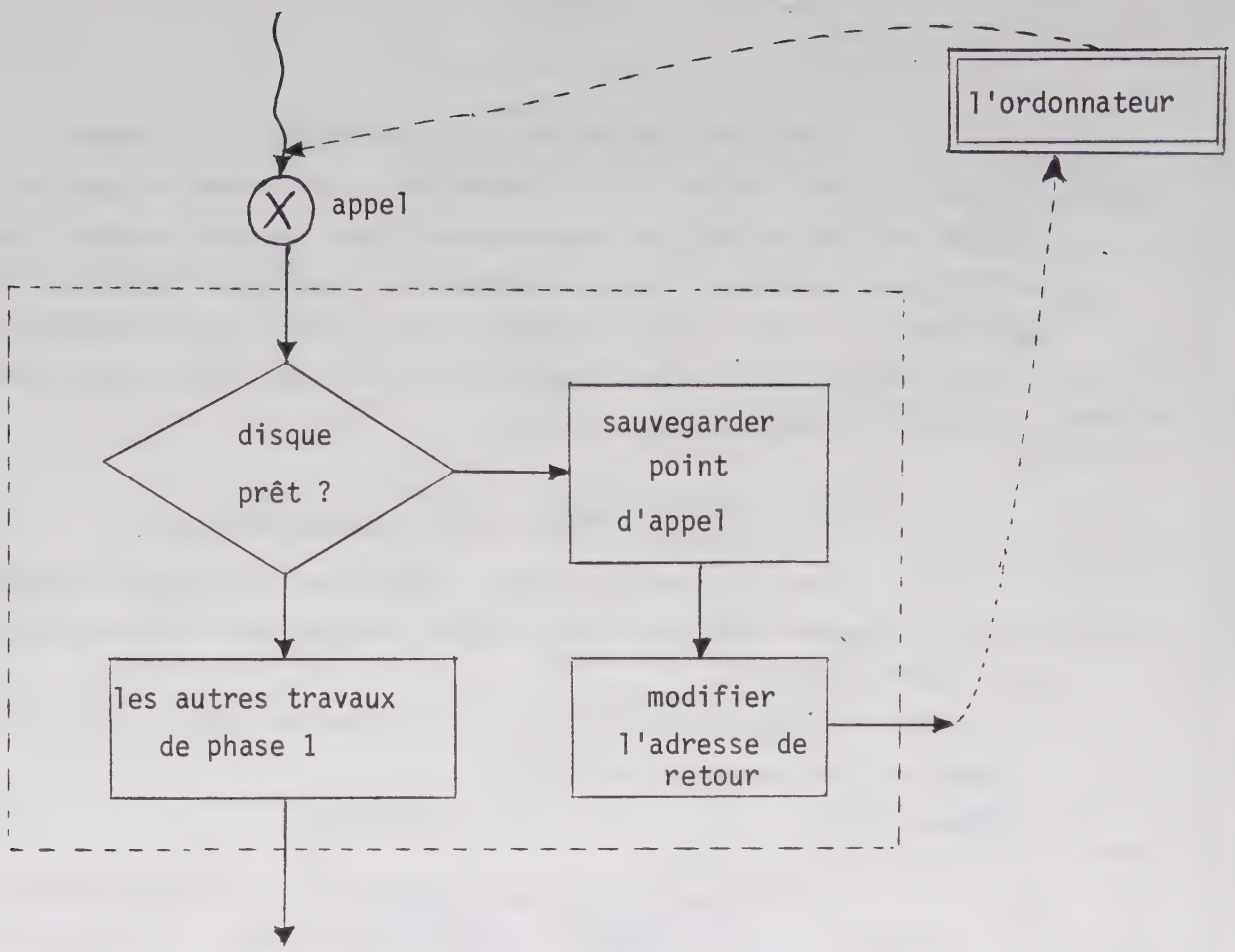


Figure 3.14 - Phase 1 d'une opération du disque

La phase 1 d'une opération du disque demande comme paramètre l'adresse absolue de la zone des données sur le disque. Mais une erreur d'adresse peut avoir de graves conséquences, aussi pour pallier cela, les programmes passeront-ils par la gestion des fichiers sur le disque pour communiquer avec la phase 1 (se référer au chapitre 5).

Le calculateur dispose d'un composant électronique BIC (Buffer Interlac Controller). Ce composant assure l'échange des informations en bloc entre la mémoire centrale et le disque. Grâce à cela, une seule interruption se produit à la fin totale de cet échange.

Si l'opération d'entrée-sortie sur le disque n'est pas réalisée complètement à cause d'une erreur matérielle, elle sera relancée un nombre fini de fois avant de positionner le code d'erreur.

2.4. Lignes 6 et 7 : télétype central

Les 2 lignes d'interruption 6 et 7 sont dédiées au télétype central. Elles servent à deux types d'opération : la lecture et l'écriture qui ont besoin comme paramètre d'une zone de la mémoire contenant les informations à échanger. L'unité d'information élémentaire d'échange entre la mémoire et le tampon du télétype est 1 caractère. Le télétype a un tampon d'entrée et un tampon de sortie, chacun a une taille de 1 caractère.

Le télétype central est une unité périphérique partagée entre les processus concurrents. La phase 1 d'une opération du télétype central doit donc s'organiser de telle manière qu'aucun processus ne soit mis en attente à l'intérieur d'elle, c'est-à-dire que sa structure ressemble à celle de la phase 1 d'une opération du disque.

En ce qui concerne les opérations d'entrée-sortie du télétype central, on s'intéresse aux caractéristiques suivantes :

- la phase 1 de lecture déclenche la lecture, mais la lecture commence réellement seulement après que l'opérateur frappe au moins une touche du clavier. On peut donc distinguer deux états : "en attente de lecture" et "en cours de lecture". Le déclenchement d'une lecture dans la phase 1, en réalité, n'est que de mettre le télétype en attente de lecture ;
- les opérations de lecture et d'écriture du télétype sont exclusives mutuellement. Pourtant, le télétype central est partageable, il est déraisonnable d'empêcher le déclenchement de l'écriture d'un processus à cause de l'attente de lecture d'un autre processus sur le télétype. Aussi, dans la phase 1 de l'écriture, si l'on trouve le télétype soit libre, soit "en attente de lecture", on reprendra le télétype pour déclencher l'écriture ; et à la fin de cette écriture, on restaurera l'état du télétype avant l'écriture (figure 3.15) ;
- la lecture du télétype central peut être appelée en même temps par plusieurs processus, mais cela entraîne la confusion des informations d'entrée. Pour éviter cette situation, les informations d'entrée doivent comporter un destinataire, donné par le numéro d'un processus (voir le chapitre "Gestion des processus concurrents"). De plus, la lecture du télétype central est organisée en deux niveaux : niveau interne et niveau externe. Le niveau interne réalise directement la lecture sur le télétype central. Il se compose des phases 1, 2 et 3. La phase 1 du niveau interne est un sous-programme de type d'entrée-sortie qui a pour but tout simple de déclencher la lecture. Seulement un processus du système a le droit d'appeler ce sous-programme. En revanche, le niveau

externe de la lecture n'est pas un sous-programme de type d'entrée-sortie. C'est un sous-programme accessible par les processus qui demandent les informations d'entrée via le télétype central.

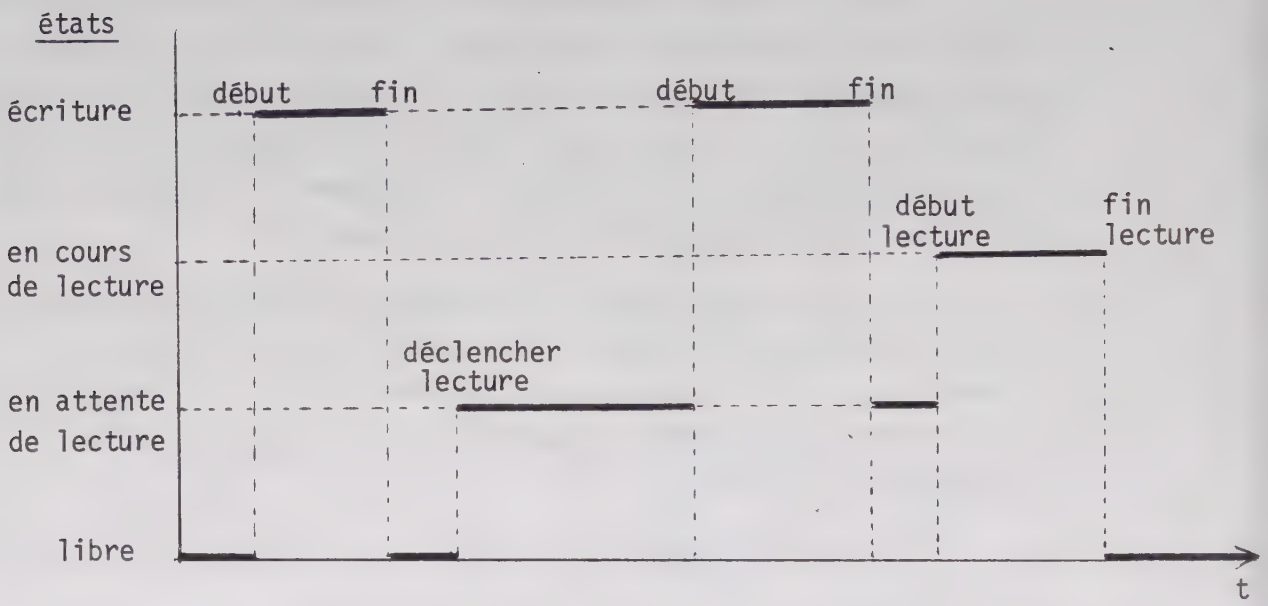


Figure 3.15 - Etats différents du télétype central

Une requête d'interruption se produit aux moments suivants :

. le tampon d'entrée du télétype devient plein. Cette requête est mise sur la ligne 7 : ligne de lecture ;
. le tampon de sortie du télétype devient vide. Cette requête est mise sur la ligne 6 : ligne d'écriture.

Pendant l'écriture du télétype, il n'y a que des requêtes d'interruption sur la ligne 6. En revanche, pendant la lecture du télétype, les informations entrées par l'opérateur, d'une part, sont chargées en mémoire, et, d'autre part, doivent apparaître au fur et à mesure sur papier du télétype. Autrement dit, la lecture entraîne une écriture passive,et elles provoquent les requêtes d'interruption sur les deux lignes. Grâce à l'exclusion mutuelle entre la lecture et l'écriture, pendant la lecture, la ligne 6 est invalidée et les requêtes d'interruption sur cette ligne sont ignorées.

2.4.1. Traitement d'interruption de la ligne 7 : phase 2 du niveau interne

La phase 2 du niveau interne de la lecture s'occupe de déposer les informations entrées dans une zone fixe de mémoire appelée "la boîte aux lettres". L'introduction éventuelle d'un caractère conventionnel("CONTROL + X") demande de recommencer la lecture à partir de début de la boîte aux lettres. La frappe sur la touche "return" du clavier (i.e., le caractère "CR") est considérée conventionnellement comme la fin totale de la lecture.

Après l'arrivée de la fin totale, la phase 2 analyse les informations dans la boîte aux lettres pour chercher un destinataire et, si elle le trouve, elle mettra une valeur non nulle dans une variable appelée "drapeau" qui est propre pour chaque processus destinataire (figure 3.16.).

2.4.2. Demande d'informations d'entrée via le télétype central

Le niveau externe de la lecture s'organise sous forme d'un sous-programme avec, comme paramètres, la longueur et l'adresse de la zone de mémoire recevant les informations entrées via le télétype central.

Le processus appelant vérifie son drapeau privé. Si la valeur du drapeau est encore nulle, il se met en attente. Ce sous-programme peut être appelé par plusieurs processus concurrents qui partagent le télétype central, il a donc le même principe de structure que la phase 1 d'une opération d'entrée-sortie sur disque (figure 3.17.).

Si la valeur de son drapeau privé est non nulle, le processus appelant copiera les informations à partir de la boîte aux lettres dans la zone de mémoire indiquée dans l'appel du sous-programme; la longueur de la copie est le minimum de la longueur réelle des informations dans la boîte aux lettres et de la longueur demandée dans l'appel de ce sous-programme.

2.5. Ligne 5 : interface asynchrone multiple

La ligne 5 est dédiée à l'interface asynchrone multiple. Actuellement, elle sert à 4 unités périphériques : les imprimantes KSR43 et Decwriter 3, le modem 300 bauds et le modem 1.200 bauds.

Une imprimante à clavier possède deux types d'opération : la lecture et l'écriture qui ont besoin comme paramètre d'une zone de mémoire contenant les informations à échanger. L'entrée et la sortie d'un modem sont associées au dialogue entre le VARIAN et l'UNIVAC à distance, via une ligne de télétransmission. Un dialogue est divisé en deux étapes : tout d'abord, le calculateur VARIAN émet

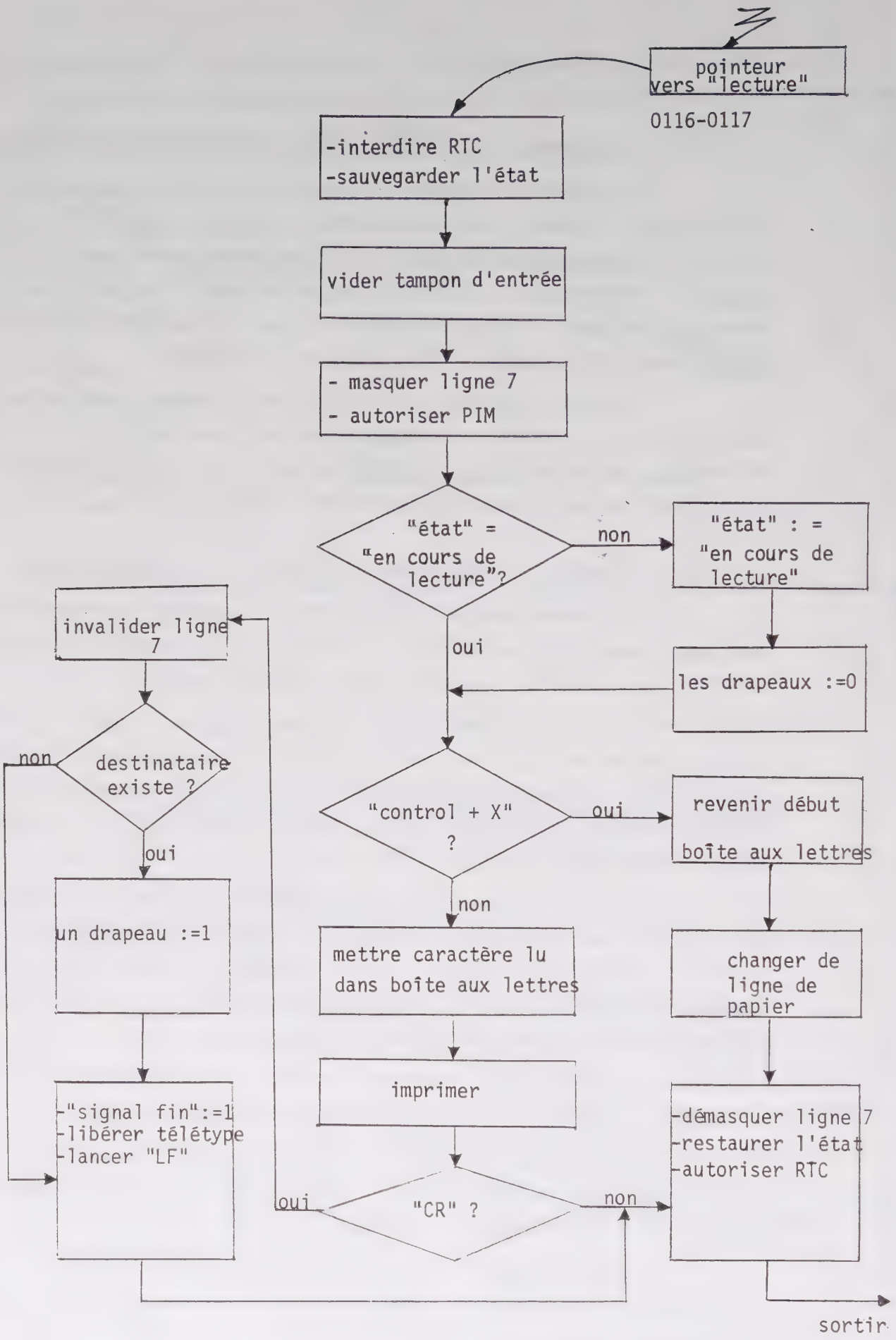


Figure 3.16.-Traitement d'interruption de la lecture du télétype central

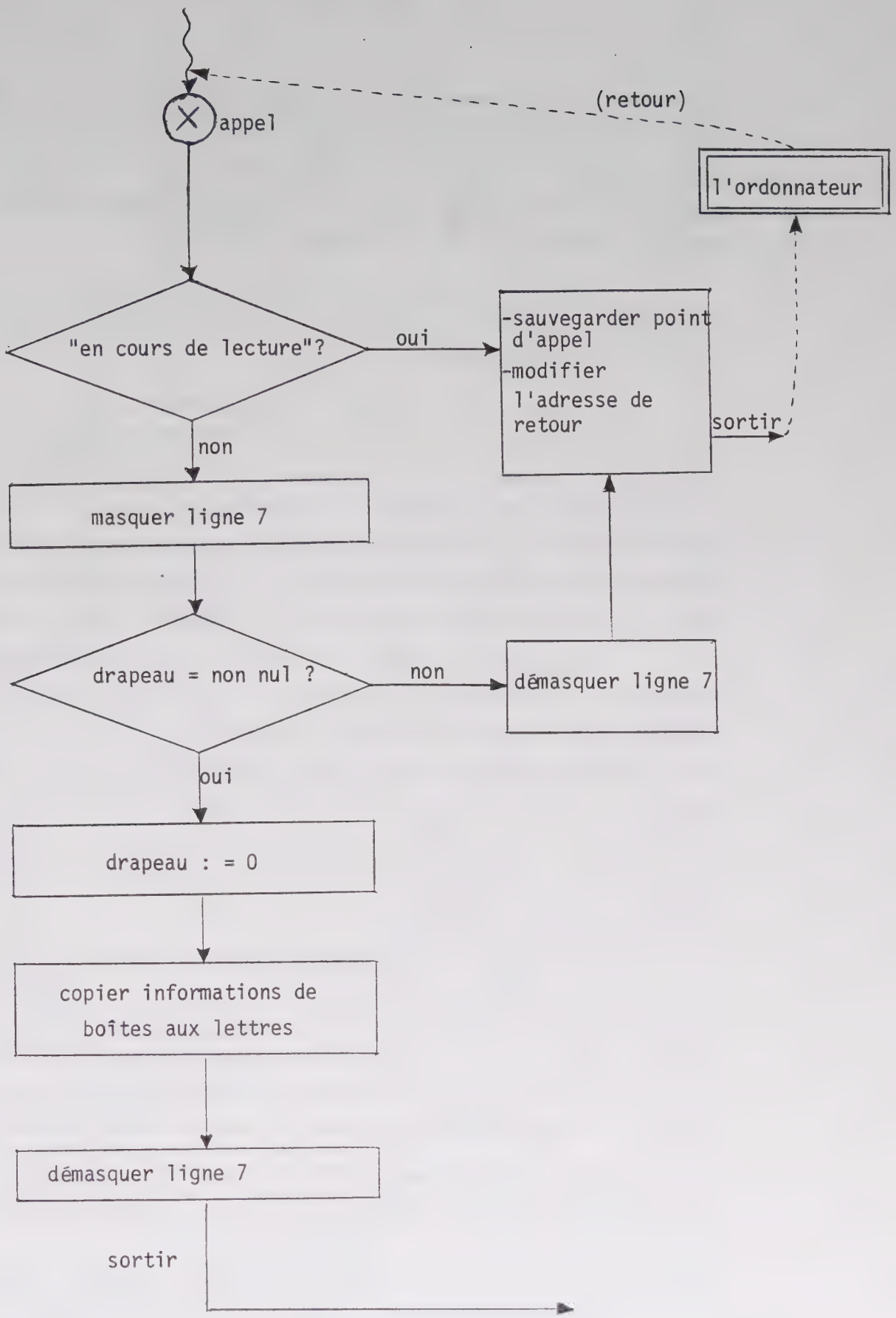


Figure 3.17. - Sous-programme de lecture du télétype central (niveau externe)

un message (ou une commande) à l'ordinateur UNIVAC, puis le VARIAN reçoit une réponse de l'UNIVAC. Aussi un dialogue a-t-il comme paramètres nécessaires : une zone de mémoire contenant les informations à émettre et une zone de mémoire contenant les informations à recevoir. Une session de conversation entre 2 machines comporte plusieurs "dialogues" séparés.

L'unité d'information élémentaire de l'échange via l'interface est 1 caractère. La taille du tampon d'entrée et du tampon de sortie de chaque imprimante et de chaque modem est 1 caractère.

Il est rappelé que quand un programmeur a besoin d'une imprimante et/ou d'un modem, il ne choisit pas exactement son imprimante ou son modem dans la liste des unités périphériques non partageables (se référer au chapitre 2). Au moment du chargement du programme en mémoire, le système désignera définitivement une imprimante (parmi deux) et/ou un modem (parmi deux) pour le programme en se basant sur la disponibilité actuelle de ces unités périphériques. Les unités désignées sont attachées à un programme jusqu'à sa terminaison.

De ce fait, la phase 1 de chaque opération d'entrée-sortie d'une imprimante ou d'un modem doit reconnaître exactement l'unité qui est attachée au programme courant pour déclencher l'opération sur cette unité. Ceci est possible grâce à une variable de l'ordonnateur appelée "ordre" et les blocs d'état des processus. La valeur de la variable "ordre" permet de connaître le numéro du programme courant (se référer au chapitre 4). Ensuite, la phase 1 consulte le bloc d'état de ce programme pour chercher les unités (imprimante et/ou modem) qui lui sont attachées.

Les unités périphériques indirectes peuvent fonctionner parallèlement, la phase 1 d'une opération sur l'imprimante ou sur le modem peut être appelée simultanément par 2 programmes concurrents (par exemple, un programme utilise le modem 300 bauds pendant qu'un autre programme utilise le modem 1.200 bauds). Cette phase 1 doit donc avoir le même principe de structure que la phase 1 d'une opération d'entrée-sortie sur disque (figure 3.18.).

Une requête d'interruption se produit aux moments suivants :

- le tampon d'entrée d'une unité devient plein : requête d'interruption de sens "entrée" ;
- le tampon de sortie d'une unité devient vide : requête d'interruption de sens "sortie".

Le fait que les unités périphériques indirectes connectées à la seule ligne d'interruption numéro 5 puissent fonctionner parallèlement nécessite que, dans l'adresse fixe attachée à la ligne 5, le pointeur vers la partie de traite-

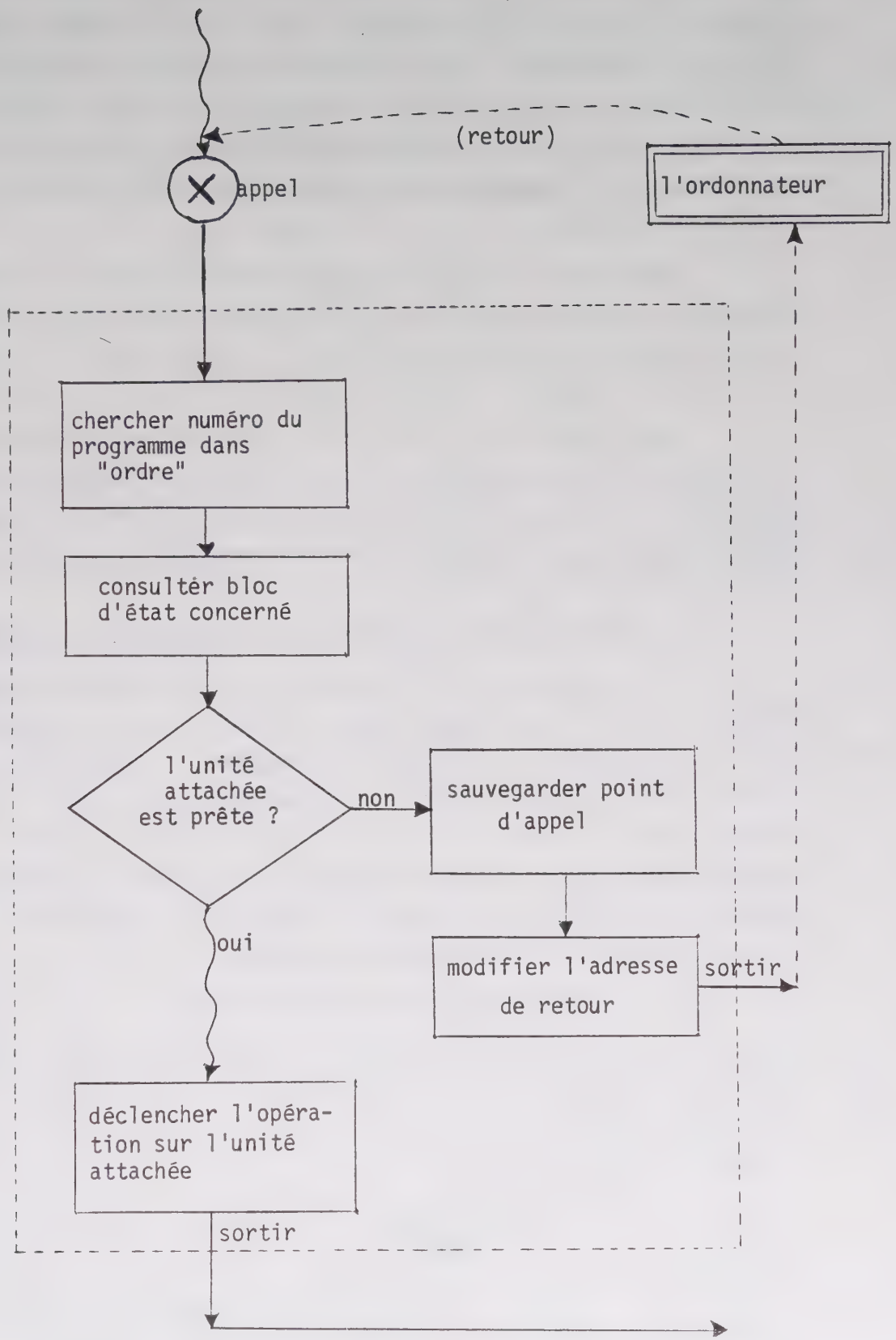


Figure 3.18.- Phase 1 d'une opération d'entrée-sortie d'une unité périphérique indirecte

ment des interruptions sur cette ligne soit toujours existant. Cette partie de traitement est donc une partie commune pour toutes les interruptions des unités périphériques indirectes. Elle se compose de plusieurs sous-parties, chacune d'elles s'occupe de traiter un type d'interruption d'une unité particulière. Lorsqu'une interruption apparaît sur la ligne 5, la partie commune doit avant tout reconnaître l'origine de cette interruption pour se brancher à la sous-partie correspondante.

Dans le cas des unités périphériques indirectes, si une requête d'interruption se produit alors qu'une autre requête est déjà en cours sur la ligne 5, cette nouvelle requête n'est pas capable de provoquer une interruption. Elle reste comme une requête en attente d'un traitement sans interruption.

Aussi, après avoir réalisé une sous-partie pour traiter une requête d'interruption, il faut vérifier s'il y a encore une requête d'interruption en attente sur la ligne 5, afin de la traiter sans attendre une nouvelle interruption sur cette ligne. Tout cela est fait en masquant la ligne 5 (figure 3.19).

2.5.1. Traitement de la lecture sur une imprimante à clavier

La lecture sur une imprimante à clavier entraîne toujours une écriture passive des caractères entrés au fur et à mesure sur papier de l'imprimante. Par conséquent, elle produit des requêtes d'interruption de sens entrée et sortie. Comme on l'a vu dans le paragraphe 1.1.3.3., pour qu'une requête d'interruption d'une unité d'un certain sens ne soit pas ignorée, une permission doit être établie dans le registre de contrôle (ACIA) de cette unité avec le même sens que celui de la requête. Pour reconnaître une requête de sens inverse, il faut changer aussi le sens de la permission.

On s'intéresse aux phénomènes suivants :

- dans la sous-partie du traitement de requête d'interruption de sens entrée (appelée sous-partie d'entrée), il faut lancer l'écriture de caractère lu. Pour reconnaître la fin de cette écriture, il faut mettre la permission de requête de sens sortie ;
- dans la sous-partie de traitement de requête d'interruption de sens sortie (appelée sous-partie de sortie), une variable permet de distinguer une requête d'interruption d'une écriture passive d'une requête d'interruption d'écriture normale. S'il s'agit de la requête d'une écriture passive, on remet la permission de requête de sens entrée pour continuer la lecture, sauf à la fin de la lecture ;
- la fin de la lecture est décidée soit par la touche "return" du clavier comme dans le cas du télétype central, soit par le dépassement du nombre de carac-

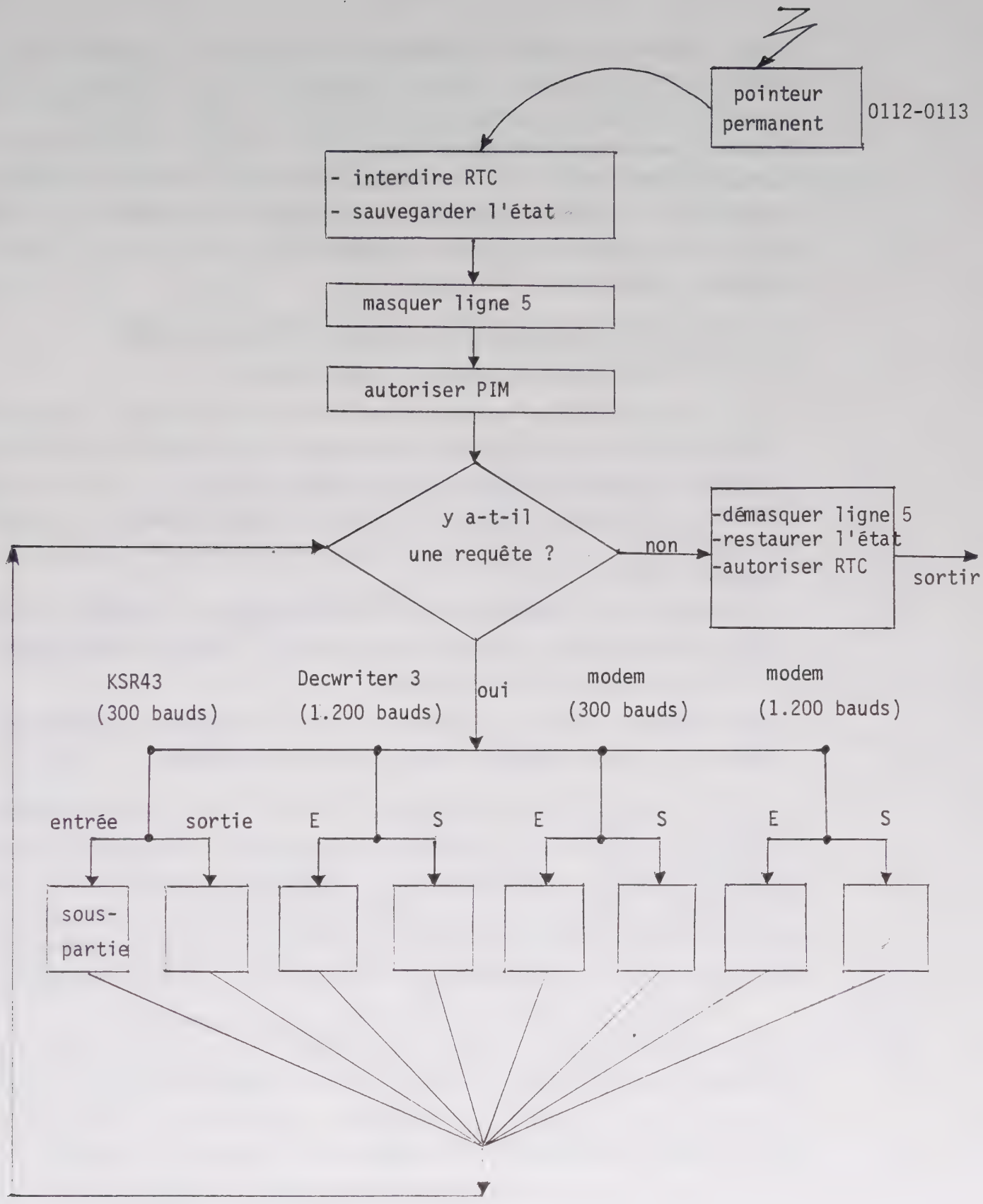


Figure 3.19.- Partie unique de traitement d'interruption de l'interface asynchrone multiple

tères d'entrée prévu dans les paramètres de la lecture. La sous-partie d'entrée signale la fin totale de la lecture, cependant, elle ne libère pas encore l'imprimante à clavier. Lorsque la sous-partie de sortie reconnaît la fin totale de la lecture, elle lance l'écriture du caractère "changement de ligne" (caractère "LF") dans le but de faciliter la reconnaissance des informations entrées sur le papier de l'imprimante. Au moment du traitement de l'interruption de cette dernière écriture, la sous-partie de sortie libère l'imprimante concernée (figure 3.20).

2.5.2. Traitement du dialogue entre le VARIAN et l'UNIVAC

Le dialogue se déroule en deux étapes :

- dans la première étape, le VARIAN joue le rôle d'un émetteur. Il émet vers l'UNIVAC de façon continue jusqu'à épuisement une chaîne de caractères qui se trouve en mémoire. Après avoir émis chaque caractère, il lui faut attendre l'écho de ce caractère (renvoyé par l'UNIVAC) avant d'émettre le caractère suivant. Le renvoi de l'écho est une caractéristique de la transmission sur la ligne de l'UNIVAC ;

  l'émission du caractère spécial "return" signale à l'UNIVAC la fin d'émission du côté du VARIAN (c'est une convention de l'UNIVAC), c'est-à-dire la fin de la première étape du dialogue ;

- dans la deuxième étape, le VARIAN reçoit les informations en provenance de l'UNIVAC. Il joue alors simplement le rôle d'un récepteur.

Du point de vue interruption, l'étape 1 de dialogue ressemble à la lecture sur imprimante à clavier, en sens inverse. L'émission d'un caractère produit une requête d'interruption de sens sortie ; en revanche, l'arrivée de l'écho de ce caractère produit une requête d'interruption de sens entrée. L'alternance des requêtes d'interruption de sens différents demande l'alternance au même rythme, du sens de permission d'interruption de l'ACIA lié au modem.

Pourtant, il y a une différence entre ce phénomène et celui de la lecture sur une imprimante à clavier : la reconnaissance d'un caractère émis et le renvoi d'un écho sont à la charge de l'ordinateur UNIVAC. Ceci permet au calculateur VARIAN d'ignorer les requêtes d'interruption de sens sortie. Aussi, pendant le dialogue le sens entrée est-il le seul sens permis.

Le signal de changement d'étape permet de reconnaître l'origine des requêtes d'interruption de sens entrée : dans l'étape 1, ce sont des échos pour vérifier l'exactitude du transfert ; par contre, dans l'étape 2, ce sont des informations à recevoir.

En ce qui concerne la fin totale du dialogue, c'est-à-dire la fin des informations en provenance de l'UNIVAC, le VARIAN est dépendant du fonction-

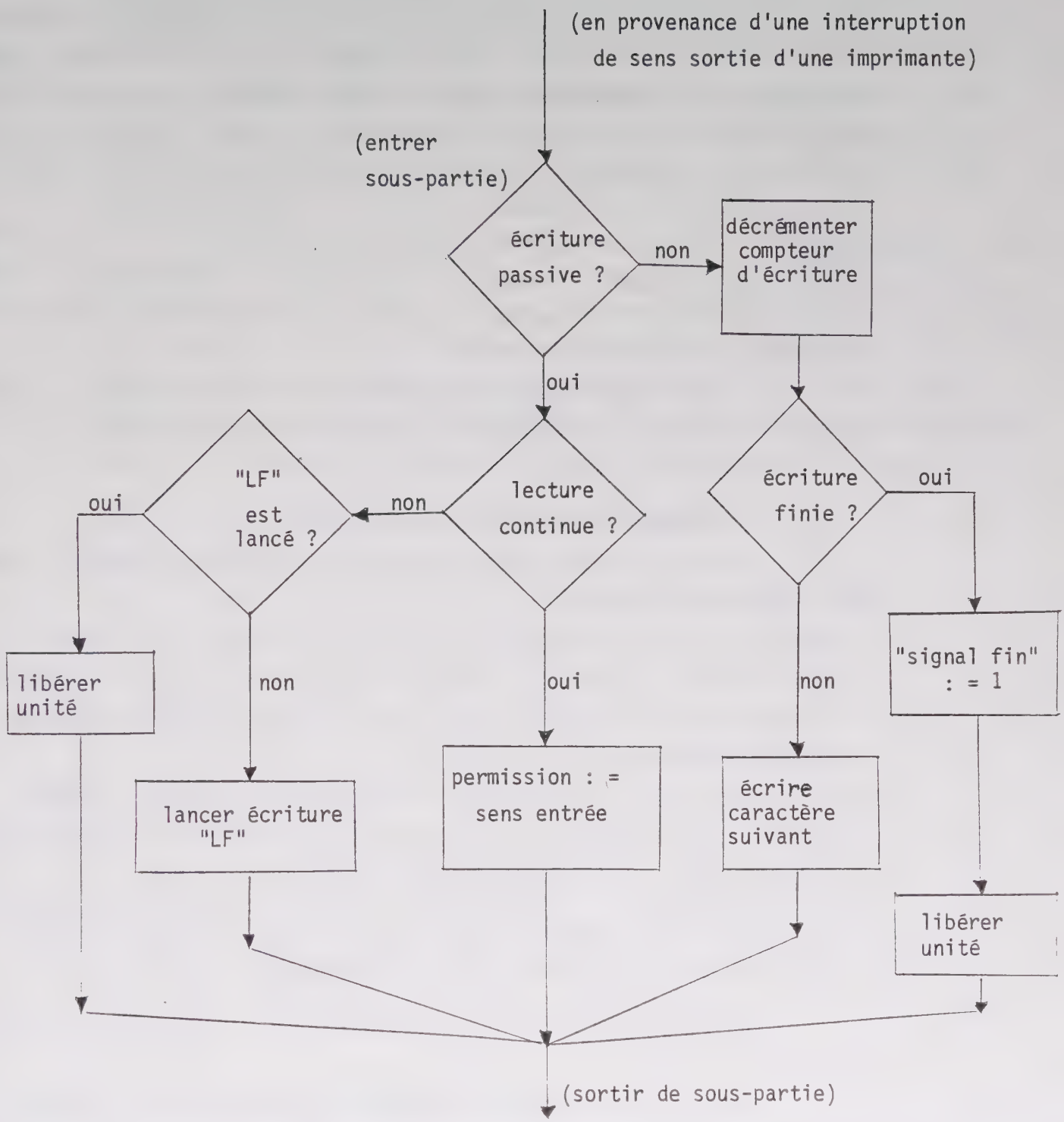


Figure 3.20.- Sous-partie de traitement d'interruption de sens sortie d'une imprimante à clavier

nement du sous-système conversationnel de l'UNIVAC. L'observation de la forme des informations en provenance de l'UNIVAC sur la ligne de télétransmission permet de voir que dans la plupart des cas la chaîne d'informations se termine par un caractère "spécial" ">", et, dans les autres cas, par le caractère "changement de ligne" (LF). Mais ces caractères peuvent également apparaître plusieurs fois à l'intérieur de la chaîne d'informations. Une seule distinction est possible : après le caractère ">" ou "LF" qui termine une chaîne d'informations, il n'y a plus rien sur la ligne ; au contraire, après le caractère ">" qui se trouve à l'intérieur de la chaîne, un autre caractère arrivera dans un intervalle de temps très bref ; quant au caractère "LF" à l'intérieur de la chaîne, un autre caractère arrivera dans les intervalles de temps très variés, parfois de l'ordre d'une dizaine de minutes.

Cette situation permet d'entrevoir une solution pour la reconnaissance de la fin totale du dialogue : intervention de l'horloge.

2.6. Traitement d'interruption de l'horloge

L'horloge participe aux résolutions concernant le temps. A présent, elle s'occupe de 3 tâches :

. elle s'associe avec l'ordonnateur pour compter le temps d'occupation du processeur de façon continue par chaque processus, afin d'annuler l'exécution d'un programme qui monopolise le processeur (voir le chapitre "Gestion des processus concurrents") ;
. elle s'associe avec la lecture d'une imprimante pour compter le temps d'attente de la réponse de l'opérateur, afin de déclencher des sonneries (voir le chapitre "Facilités d'exploitation") ;
. elle s'associe avec le dialogue entre le VARIAN et l'UNIVAC sur la ligne de transmission pour reconnaître la fin totale du dialogue. Dans cette association, l'horloge réalise un travail assez complexe.

Dans ce chapitre, on ne s'intéresse qu'à la troisième tâche de l'horloge.

Il y a un lien entre le traitement d'interruption de l'horloge et le traitement d'interruption du modem au travers des variables communes : un compteur de temps d'attente et la zone de mémoire contenant les informations reçues via le modem.

Le traitement du dialogue dans l'étape 2 indique au traitement d'interruption de l'horloge le caractère non nul reçu le plus récent, et, en même temps, remet le compteur d'attente à zéro. Si le caractère reçu est ">", la

partie de traitement d'interruption de l'horloge incrémente le compteur d'attente et, si ce compteur dépasse une valeur prévue, le traitement de l'horloge signalera la fin totale de dialogue et libérera le modem concerné.

Cette politique ne peut pas être appliquée pour le cas de caractère "LF" parce que le temps d'attente dans ce cas doit être prévu par une valeur assez grande, ce qui empêche le déroulement normal du programme appelant le dialogue.

On note que la dernière ligne des chaînes d'informations qui se terminent par "LF" appartient à un groupe de formes standard. Cette observation donne une solution : lorsque le traitement d'interruption de l'horloge reconnaît un caractère "LF" reçu par un modem, il compare la dernière ligne reçue avec les formes standard (*OK*, *BRK*, *NO RUN ACTIVE*, etc..) et, si cette ligne ressemble à une forme standard, le traitement d'interruption de l'horloge signalera la fin totale de dialogue et libérera le modem concerné (figure 3.21.).

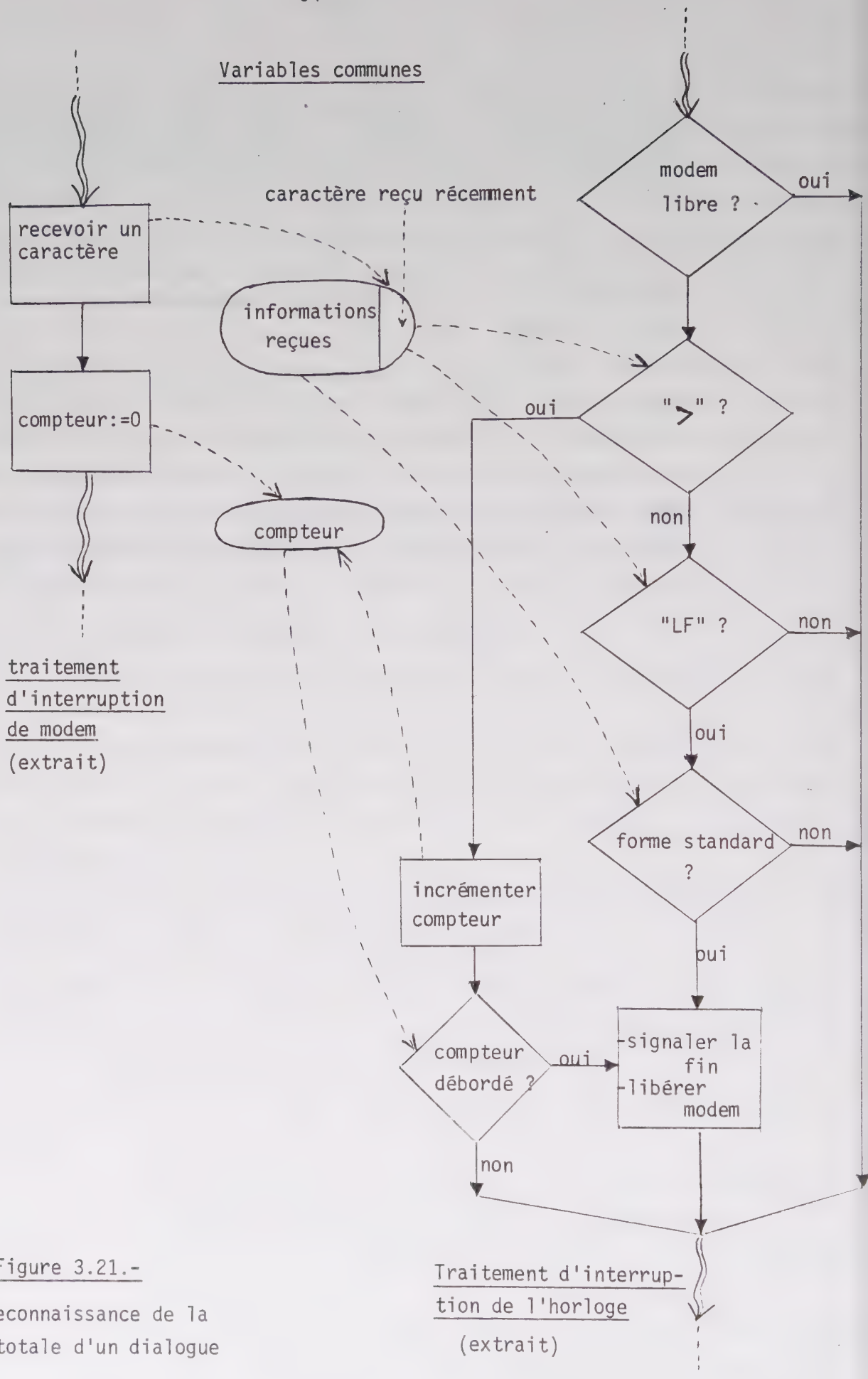


Figure 3.21.-

La reconnaissance de la fin totale d'un dialogue

C H A P I T R E IV

GESTION DES PROCESSUS CONCURRENTS

1 - PROCESSUS CONCURRENTS

Le but du projet est de réaliser un système qui permet d'éxécuter simultanément 2 programmes d'application. Ce sont 2 processus programme coexistants en mémoire.

Naturellement, il faut aussi un processus qui aide l'opérateur à diriger le fonctionnement du calculateur. C'est un processus du système.

De plus, comme on l'a vu dans la gestion des entrées du télétype central (chapitre 3), pour éviter la confusion des informations entrées via le télétype central selon les demandes de 3 processus précédents, il faut un processus qui s'occupe de recevoir et de distribuer ces informations aux processus concernés.

Au total, on peut avoir 4 processus qui s'éxécutent concurremment : 2 processus programme et 2 processus système. Ces processus sont numérotés de 0 à 3 : le numéro 0 est destiné au processus qui s'occupe de recevoir et de distribuer les informations entrées via le télétype central, le numéro 1 est destiné au processus qui éxécute des commandes de l'opérateur, les numéros 2 et 3 sont destinés aux processus programme, respectivement chargés dans la partie basse et la partie haute des adresses en mémoire.

Enfin , il faut un gérant de ces processus concurrents : c'est une partie du système appelée "l'ordonnateur" qui joue le rôle d'un arbitre pour distribuer le temps du processeur (que l'on peut appeler simplement le contrôle) aux processus.

1.1. Caractéristiques des processus -

1.1.1. Processus numéro 0

Le processus système numéro 0 (que l'on peut appeler simplement le

processus 0) a seul le droit de déclencher la lecture sur le télétype central. Il déclenche une lecture, puis il se met en attente des informations introduites par l'opérateur. Les informations reçues, auxquelles est associé obligatoirement le numéro d'un processus, sont déposées dans un tampon appelé "boîtes aux lettres". L'entrée d'un caractère conventionnel (le caractère "return") signale la fin d'une lecture. Le processus 0 analyse les informations reçues, affiche le destinataire au moyen de "drapeaux". Il y a 3 drapeaux correspondant aux 3 processus 1, 2, 3. Le contenu non nul d'un drapeau indique la présence d'un message envoyé au processus concerné dans la boîte-aux-lettres. En cas d'absence de destinataire dans les informations entrées, aucun processus n'est alors concerné. Enfin, le processus 0 déclenche de nouveau la lecture sur le télétype central.

Le processus 0 est donc un processus permanent. Il se trouve dans la zone mémoire du système. De par son fonctionnement, le processus 0 ne dispose que du télétype central, de plus, il communique avec celui-ci seulement dans le sens "entrée" (demande d'informations).

Actuellement, en tenant compte du fait que le débit des informations entrées via le télétype central est très faible, la boîte-aux-lettres est un tampon d'un seul message; chaque lecture sur le télétype central modifie les drapeaux et la boîte-aux-lettres sans vérifier si les informations précédentes ont été prises par son destinataire.

1.1.2. Processus numéro 1

Le processus système numéro 1 (que l'on peut appeler simplement le processus 1) s'occupe d'éxécuter les commandes de l'opérateur, l'une après l'autre. Il attend une commande qui devra lui être envoyée dans la boîte-aux-lettres . Lorsque une commande apparaît, le processus 1 éxécute une partie du système correspondant à cette commande, puis il signale à l'opérateur l'achèvement de cette éxécution. Enfin, il se met en attente de nouveau d'une autre commande.

Le processus 1 est donc un processus permanent. Il se trouve dans la zone mémoire du système.

Les commandes de l'opérateur ont pour but de diriger le fonctionnement du calculateur et de faire également quelques travaux divers. Parmi ces commandes on peut citer la commande du chargement d'un programme en mémoire et la commande de la suppression de l'éxécution d'un programme. L'éxécution des commandes principales sera étudiée plus loin dans ce chapitre.

Le fonctionnement du processus 1 fait qu'il ne dispose que des unités périphériques partageables. De plus, il communique avec le disque seulement

dans le sens "entrée" et il communique avec le télétype central seulement dans le sens "sortie". Pourtant, le processus 1 aura la possibilité d'accéder à quelques unités périphériques non-partageables en cas de nécessité.

1.1.3. Processus programme

Un processus programme (que l'on peut appeler simplement un programme) est introduit grâce à une commande de l'opérateur. Il se trouve dans la zone mémoire réservée pour les programmes. Il a pour but de réaliser un service ou une application spécifique. En général, sa tâche s'achèvera dans un temps fini. Un programme n'existe donc pas en permanence dans la mémoire.

Un programme dispose exclusivement des unités périphériques non-partageables qui figurent dans la liste donnée dans l'en-tête du programme. De plus, il partage avec les autres processus le télétype central et le disque. Un programme accède au disque à travers une gestion des fichiers sur disque. Il communique avec le télétype central seulement dans le sens "sortie". Pour les informations entrées via le télétype central, il va les chercher dans la boîte-aux-lettres.

La communication entre les processus et les unités péripéhriques partageables est donnée par la figure 4.1.

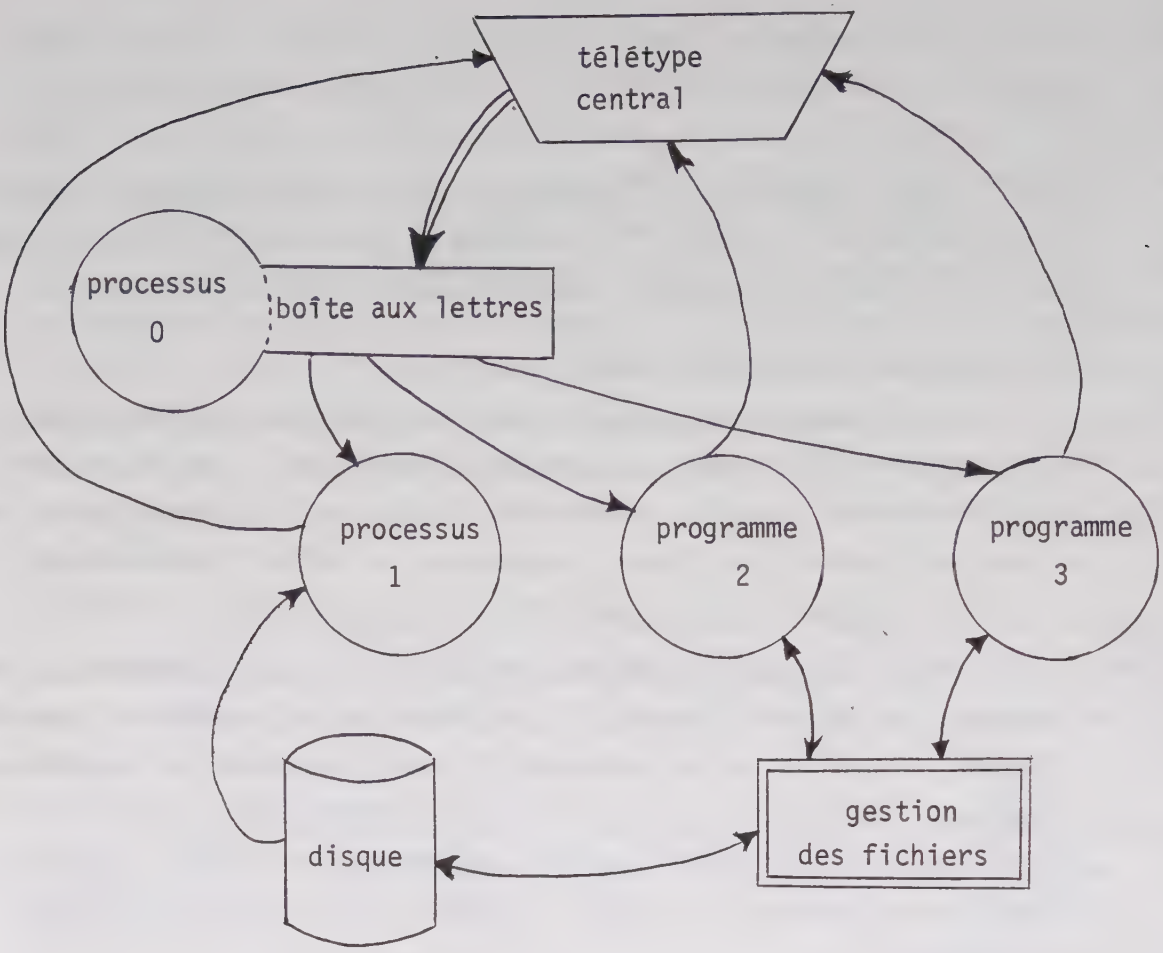


Figure 4.1.- L'accès des processus au disque et au télétype central

1.2. Etat du processus -

Le contrôle du calculateur est distribué aux processus concurrents. Chaque processus n'est pas capable d'occuper le contrôle pendant toute sa vie en mémoire. Il est obligé d'abandonner provisoirement le contrôle dans les cas suivants :

- il se met en attente de la fin d'une opération d'entrée-sortie qu'il a lancé lui-même.
- il a besoin de lancer une opération d'entrée-sortie mais l'unité périphérique concernée est encore occupée par une autre opération d'entrée-sortie.

Chaque processus a un bloc d'état pour sauvegarder son état lorsqu'il abandonne le contrôle. L'état d'un processus est représenté par l'image du compteur ordinal et le contenu des registres. Une valeur non nulle dans l'image du compteur ordinal dans un bloc d'état indique l'existence du processus

correspondant dans la mémoire.

Pour les processus permanents du système (les processus 0 et 1), ces valeurs sont donc toujours non nulles; au contraire, pour un programme (numéro 2 ou 3), l'image de son compteur ordinal doit avoir la valeur nulle pendant le temps où ce processus n'existe pas en mémoire. L'image du compteur ordinal d'un programme est donc initialisée à zéro.

Le bloc d'état d'un programme contient également les renseignements suivants :

- le nom du programme
- la position du programme en mémoire
- les unités périphériques non-partageables qui sont réservées au programme

2 - FONCTIONNEMENT DE L'ORDONNATEUR

L'ordonnateur s'occupe de distribuer le contrôle aux processus. Il dispo d'une variable appelée "ordre" contenant le numéro du processus en cours, et l'ordonnateur a seul le droit de modifier le contenu de cette variable.

Quand un programme abandonne le contrôle, il redonne le contrôle à l'ordonnateur. Dans le cas d'un abandon provisoire, l'ordonnateur sauvegarde l'état du processus se mettant en attente dans son bloc d'état. Ensuite, il change le contenu de la variable "ordre" et examine l'image de compteur ordinal dans le bloc d'état du processus dont le numéro est dans "ordre". Si cette image de compteur ordinal n'est pas nulle, c'est-à-dire que ce processus existe, l'ordonnateur restaurera l'état de ce processus à partir de son bloc d'état et lui redonnera le contrôle. Si le processus examiné n'existe pas, l'ordonnateur change de nouveau le contenu de "l'ordre" et examine l'état du processus suivant. (figure 4.2).

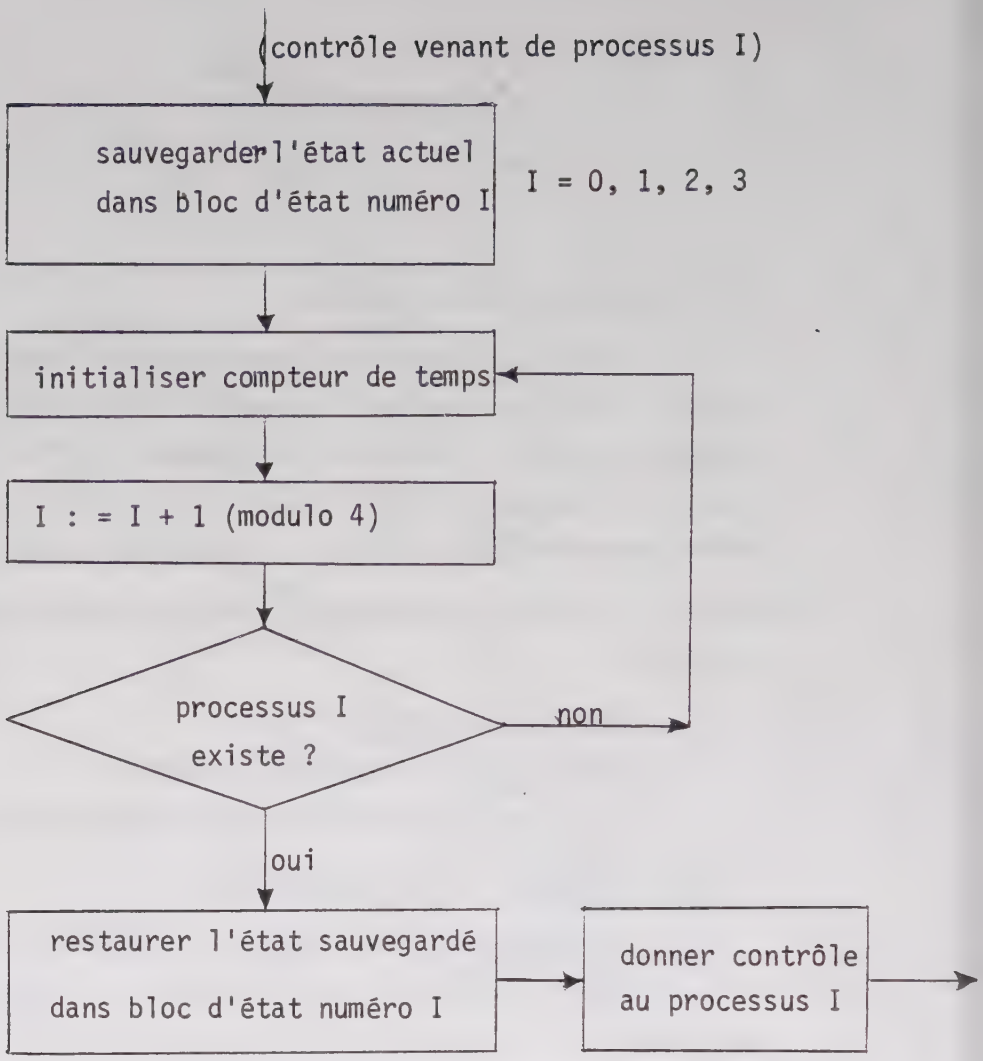


Figure 4.2. L'ordonnateur

Un changement du contenu de l'"ordre" est en fait une incrémentation modulo 4, c'est-à-dire que l'ordonnateur distribue le contrôle aux processus de façon cyclique, avec le principe "chacun son tour", aucun processus n'étant plus prioritaire que les autres. Cependant, les intervalles de temps du processeur distribués aux processus ne sont pas égaux. Ils dépendent du déroulement interne de chaque processus : attente de disponibilité d'une unité périphérique ou attente de fin totale d'une opération d'entrée-sortie.

Il est à noter que les interruptions, quelles qu'elles soient, ne provoquent pas l'alternance de processus. Pourtant, on prévoit le cas où un programme tombe dans une boucle infinie sans opération d'entrée-sortie, alors, il ne redonne jamais plus le contrôle à l'ordonnateur, c'est-à-dire qu'il monopolise le processeur et bloque tous les autres processus. Ce phénomène est la conséquence d'une erreur de programmation (dans le plupart des cas) ou d'un

conflit insoluble sur des fichiers sur disque (voir le chapitre "Gestion des fichiers sur disque"). Evidemment, ce phénomène ne peut pas se produire pour les processus système.

Pour interrompre un programme en boucle infinie, on utilise une solution qui nécessite un lien entre l'ordonnateur et le traitement d'interruptio de l'horloge. Ce lien est un compteur de temps. L'ordonnateur remet à une valeur initiale ce compteur avant de donner le contrôle à un processus, alors que le traitement d'interruption de l'horloge incrémente ce compteur. Si ce compteur arrive à une valeur limite, c'est que le programme en cours monopolise le contrôle depuis longtemps.

Le traitement d'interruption de l'horloge incrémente et analyse ce compteur, et lorsqu'il trouve que ce compteur dépasse la valeur limite, il force le programme en cours à se brancher à la procédure de terminaison au moyen d'une modification de l'adresse de retour; ainsi, ce programme se termine involontairement, et par conséquent, redonne le contrôle à l'ordonnateur (figure 4.3.).

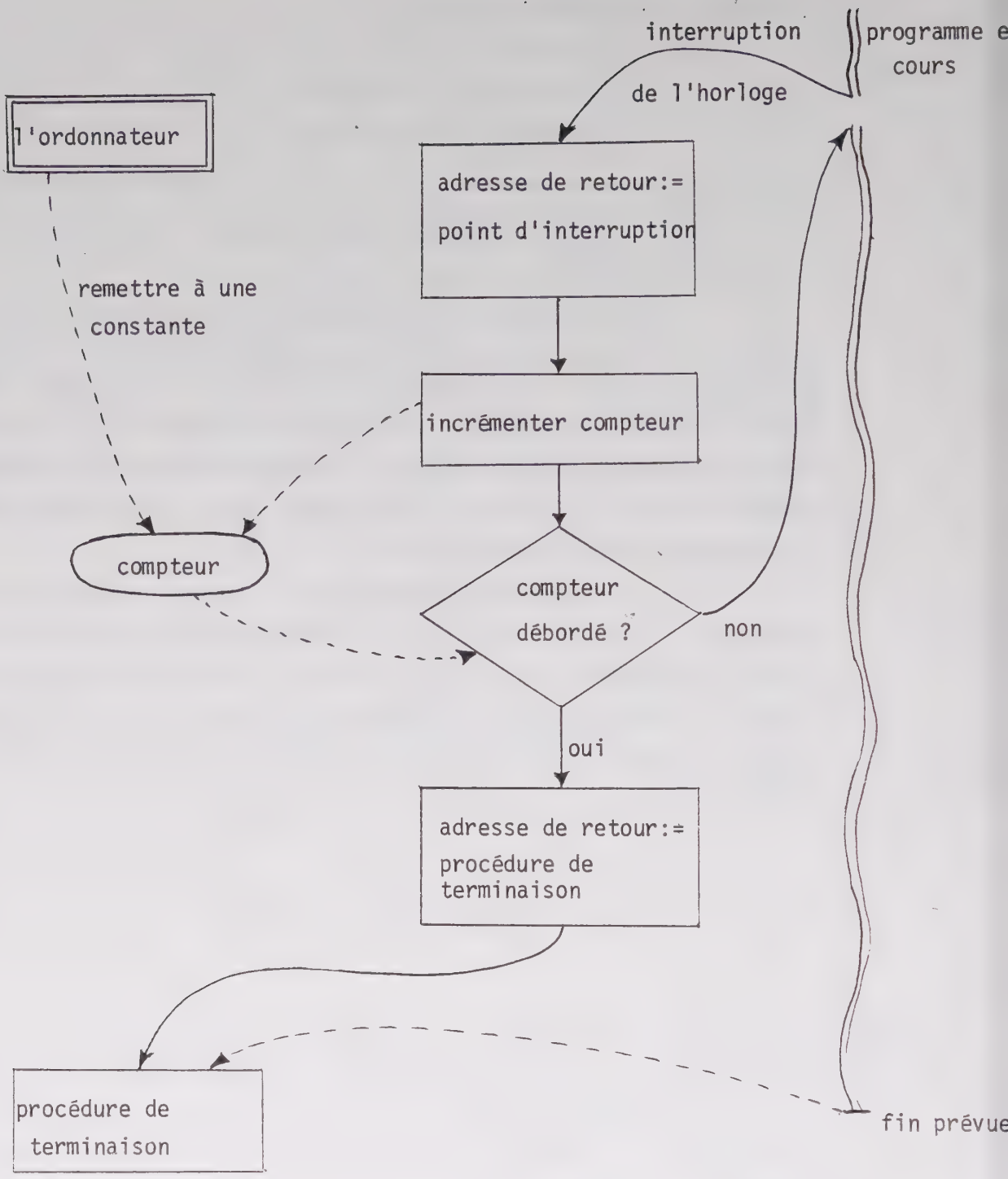


Figure 4.3. L'intervention de l'horloge dans la gestion des processus

3 - EXECUTION DES COMMANDES

L'opérateur dirige l'exploitation du calculateur par des commandes. L'éxécution d'une commande est l'éxécution d'une partie correspondante du système. L'ensemble des commandes est vaste. On ne présentera ici que les commandes principales concernant la gestion des processus concurrents, telles que le chargement d'un programme en mémoire, la suppression de l'éxécution d'un programme et la mise en place des clés logiques. Ces commandes permettent à l'opérateur d'intervenir dans l'éxécution des programmes pour des conditions spécifiques.

3.1. Chargement d'un programme en mémoire -

Les modules chargeables sont stockés sur un support (disque ou bande). Le chargement d'un programme en mémoire est réalisé en plusieurs phases, telles que la recherche de l'en-tête du programme sur le support, l'analyse des informations dans l'en-tête du programme et l'état actuel du calculateur pour savoir si le chargement est possible, puis le chargement proprement dit et la modification de la réservation des ressources après le chargement (figure 4.4.).

3.1.1. Réservation des ressources

La partie du système appelée "le chargeur" s'occupe du chargement. Il s'occupe de gérer la table correspondant à la réservation des ressources : mémoire centrale et unités périphériques non-partageables.

Le descripteur de la mémoire contient 2 pointeurs dynamiques qui pointent l'un vers la fin du corps du programme dans la position numéro 2, l'autre vers le début du programme dans la position numéro 3 (voir la figure 2.3). La différence entre les valeurs de ces 2 pointeurs donne la taille de mémoire disponible. La taille de la mémoire nécessaire pour charger un programme est la somme de la longueur de son corps et de la longueur de son descripteur des adresses translatables. Dans le choix de la position en mémoire pour un programme à charger, la position numéro 2 est examinée toujours avant la position numéro 3. Cet ordre pour l'examen donne la possibilité de récupérer la partie de la mémoire occupée par le descripteur des adresses translatables du programme

Quant aux unités périphériques, le chargeur possède 2 listes des unités périphériques non-partageables occupés par 2 programmes (voir la figure 2.2.). Pour un programme n'existant pas en mémoire, cette liste est évidemment vide. Le chargeur compare la liste correspondant au programme à charger avec la liste du programme existant en mémoire, et s'il y a un conflit entre elles, le chargement sera refusé. Si la liste commandée est acceptée, le chargeur vérifie l'état

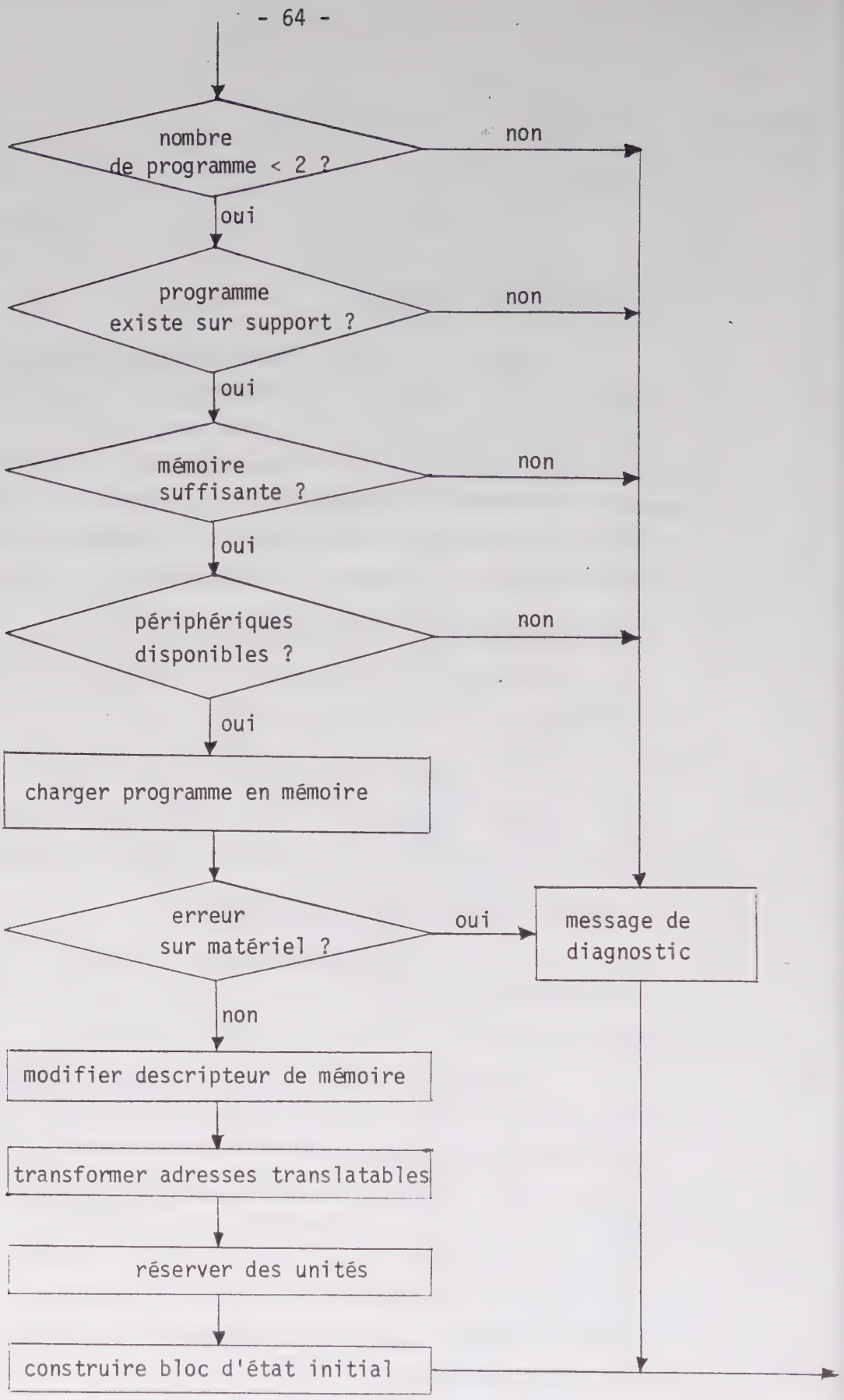


Figure 4.4. Chargement d'un programme

des unités périphériques figurant dans cette liste pour savoir si elles sont disponibles physiquement (i.e. si elles sont connectées au calculateur et si elles ont été démarrées).

En ce qui concerne les modems, le chargeur cherche à désigner pour le programme demandeur un modem qui est disponible physiquement et ne figure pas dans la liste du programme voisin existant. L'examen est réalisé toujours sur le modem 300 bauds avant le modem 1200 bauds. La méthode de désignation d'une imprimante à clavier est analogue. L'examen est réalisé toujours sur le KSR43 avant le Decwriter 3. Cet ordre dans l'examen des modems et des imprimantes à clavier est simplement une convention tenant compte de l'exploitation réelle du calculateur.

La modification des pointeurs du descripteur de la mémoire et la mise en place de la liste des unités périphériques du programme à charger ne sont faites réellement qu'après le transfert complet du programme à partir d'un support.

3.1.2. Introduction d'un nouveau processus -

Lorsque le programme est chargé, le chargeur construit un bloc d'état en consultant les informations dans l'en-tête du programme sur le support. L'adresse relative de l'instruction de début d'éxécution du programme par rapport au début du module devient maintenant une adresse absolue. Elle est mise dans ce bloc d'état.

Le chargeur fonctionne sous le contrôle du processus 1. Lorsqu'il se termine, le processus 1 se met en attenre d'une nouvelle commande et le contrôle sera redonné à l'ordonnateur. Celui-ci va trouver un nouveau processus candidat et lui donner le contrôle.

3.2. Supression de l'éxécution d'un programme -

Il se peut que l'opérateur soit obligé d'arrêter l'éxécution d'un programme à un certain moment, dans ce cas, il utilise cette commande.

En général, le déroulement normal d'un programme amènera à une fin normale prévue où le programme se branchera volontairement à la procédure de terminaison pour qu'il abandonne enfin définitivement le contrôle. Or, l'éxécution de cette commande modifie l'image du compteur ordinal dans le bloc d'état du programme en question en y mettant une adresse correspondant à la procédure

de terminaison. Lorsque ce programme reprend le contrôle, il se branchera immédiatement à cette procédure, et ainsi rendra le contrôle.

3.3. Mise en place des clés logiques -

Comme on l'a vu dans le chapitre "INTRODUCTION", le calculateur dispose de clés physiques sur son panneau, et il existe des instructions concernant l'état de ces clés. La mise de l'état des clés physiques est réalisée manuellement. Grâce aux clés physiques, l'opérateur est capable d'intervenir dans l'éxécution d'un programme.

Or, il est impossible de se servir d'un seul jeu de clés physiques pour deux programmes indépendants s'éxécutant simultanément. De ce fait, chaque programme dispose à présent d'un mot en mémoire contenant l'image des clés physiques qui lui sont associées. On l'appelle mot "clés logiques" du programme.

Chaque clé logique est représentée par un bit, et la valeur de ce bit représente l'état de la clé logique correspondante. La mise en place des clés logiques pour un programme correspond à affecter le mot "clés logiques" lui appartenant avec une valeur déterminée.

Les instructions concernant l'état de ces clés physiques sont invalidées et remplacées par un sous-programme unique qui examine les clés logiques.

Le changement manuel des clés physiques sur le panneau est maintenant remplacé par l'éxécution d'une commande avec une valeur choisie.

L'opérateur conserve donc toujours la possibilité d'intervenir dans l'éxécution de chaque programme.

3.4. Suppression de l'éxécution d'une commande

L'éxécution d'une commande est l'éxécution d'une partie du système, y compris les opérations d'entrée-sortie. Une opération d'entrée-sortie entraîne fréquemment des attentes. Une commande en cours d'attente empêche l'éxécution des autres commandes. Il se peut également que l'éxécution d'une commande soit bloquée à cause d'une anomalie d'une unité périphérique concernée. On a besoin donc de temps en temps de supprimer la commande en cours pour pouvoir lancer une autre commande. Pour cela, on se sert du processus 0.

Dans les informations entrées via le télétype central, le numéro de processus est alors remplacé par des caractères spéciaux conventionnels.

Pendant la phase d'analyse des informations reçues, le processus 0 reconnait ces caractères, il remplace l'image du compteur ordinal dans le bloc d'état du processus 1 par l'adresse de fin d'éxécution de la commande. Lorsque le processus 1 reçoit le contrôle, il se met immédiatement en attente d'une nouvelle commande.

Il existe donc une influence mutuelle entre les processus gérés par l'ordonnateur. Ce phénomène est donné par la figure 4.5.

4 - TERMINAISON D'UN PROGRAMME

Un programme se termine dans un des trois cas suivants :

. Fin prévue du programme
. Supression par une commande de l'opérateur
. Suppression par intervention de l'horloge

Dans tous les cas, le programme se branche à la procédure de terminaison

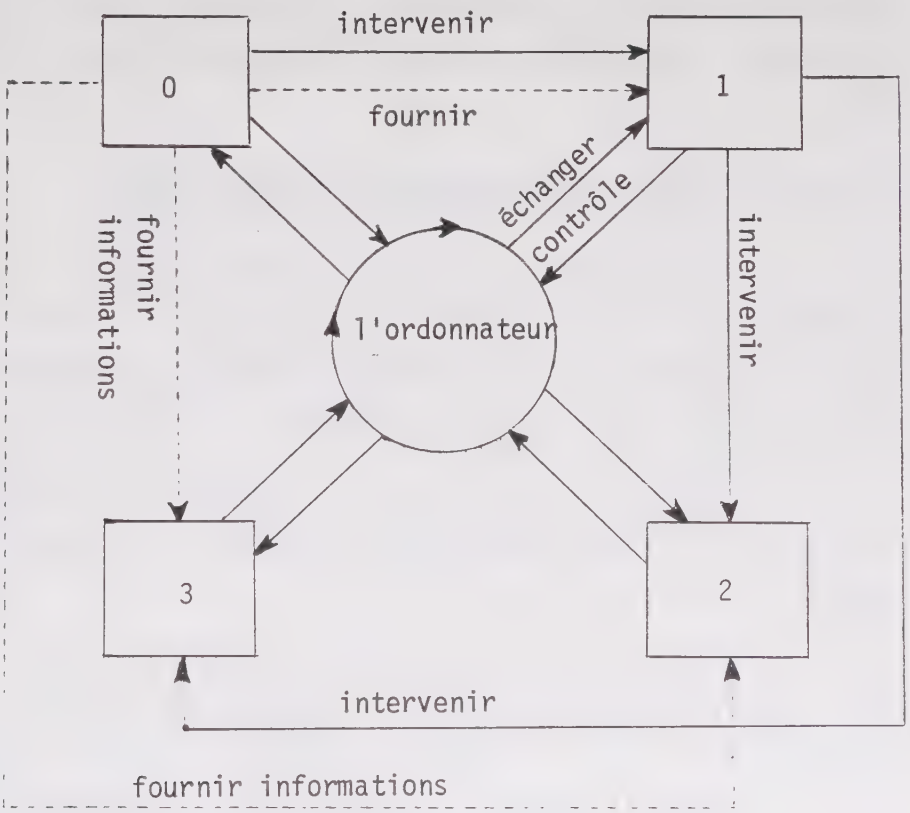


Figure 4.5. Relations mutuelles des processus

Ceci est volontaire en cas de fin prévue, par contre, il est forcé dans les autres cas.

La procédure de terminaison est une partie résidante du système qui s'occupe de résoudre tous les problèmes après la fin d'un programme. Après avoir sorti un message donnant la cause de la fin, elle modifie la réservation des ressources.

Un des deux pointeurs du descripteur de la mémoire est modifié. Si le programme 2 se termine, le pointeur concerné pointera vers la fin de la partie A du système; si le programme 3 se termine, le pointeur concerné pointera vers le début de la partie B du système.

Ensuite, cette procédure consulte la liste des unités périphériques non-partageables attachées au programme afin de les remettre à l'état libre au moyen de modification des variables "état" correspondantes (voir chapitre 3). Puis, la liste des unités périphériques dans le bloc d'état du programme terminé est remise à 0.

Le programme redonne le contrôle à l'ordonnateur en préparant l'adresse zéro pour le retour. L'ordonnateur sauvegardera cette valeur zéro dans l'image du compteur ordinal dans le bloc d'état du programme, alors ce programme n'existe plus.

5 - ASPECT GLOBAL DU FONCTIONNEMENT DU CALCULATEUR

Dans le chapitre 3, avec la gestion des opérations d'entrée-sortie par des interruptions élémentaires, on a vu que le processeur et les unités périphériques peuvent fonctionner de façon parallèle. A présent, avec la gestion des processus concurrents, on peut illustrer la distribution de temps du processeur par la figure 4.6.

Les intervalles de temps du processeur distribués aux processus sont découpés par les traitements des interruptions en provenance des unités périphériques et de l'horloge.

De plus, pour avoir un aspect global du fonctionnement simultané des processus , on le modélise par un réseau de Pétri marqué (se référer à l'article "Petri Nets" de James L. Peterson, Computing Surveys, Septembre 1977).

Un réseau de Pétri est un graphe qui se compose de 2 types de sommets :

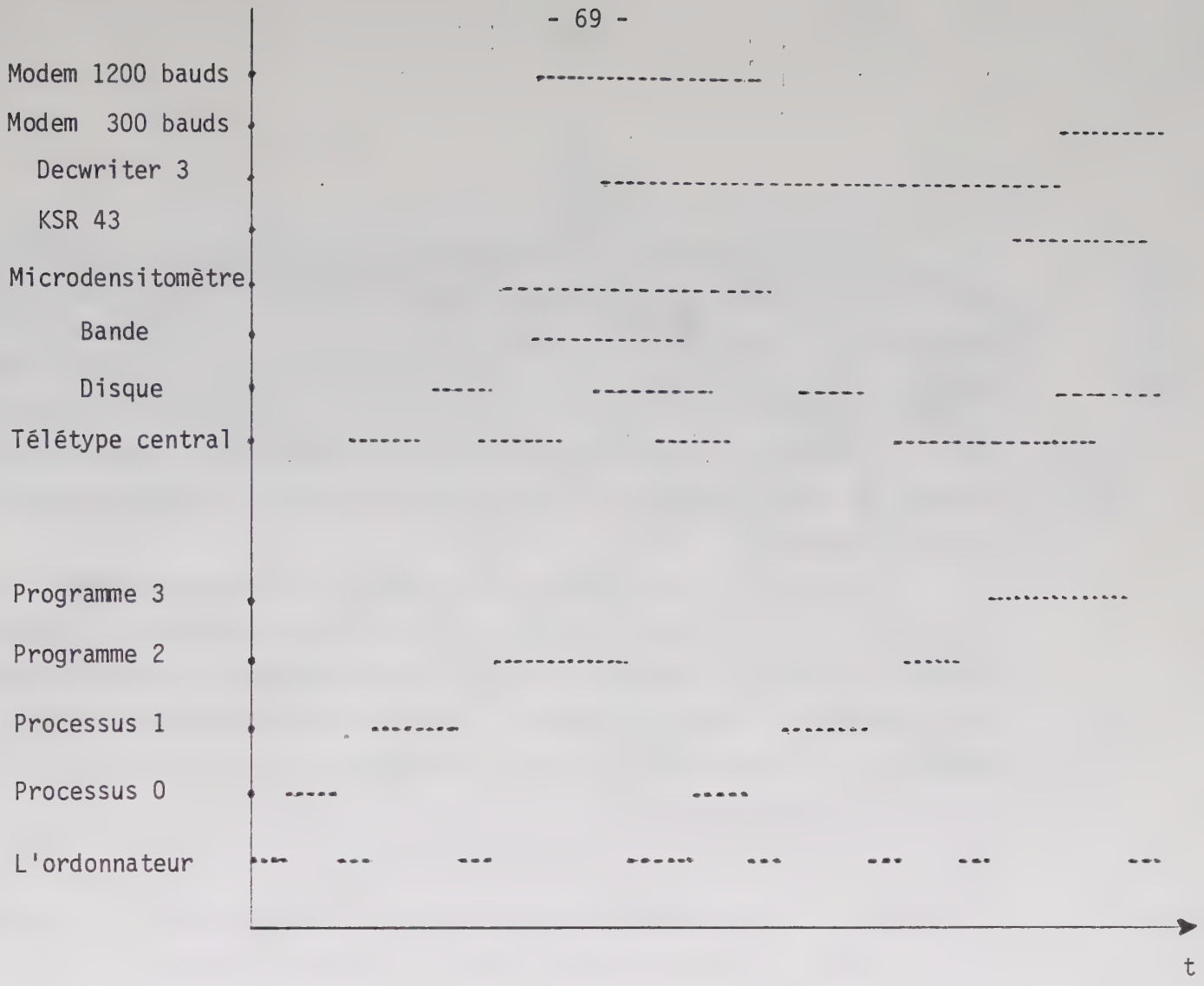


Figure 4.6. Distribution de temps du processeur

les cercles (○) (appelé "place") et les barres (//) (appelée "transition"). Les arcs orientés connectent les places avec les transitions. Une place connectée à une transition par un arc orienté vers cette transition est appelée une place d'entrée de cette transition (○→//). Une place connectée à une transition par un arc orienté vers cette place est appelée une place de sortie de cette transition (//→○).

En plus des propriétés statiques représentées par un graphe, un réseau de Pétri aura des propriétés dynamiques lorsque des "marques" lui sont ajoutées. Ce sont des jetons, représentés par des gros points noirs, qui se trouvent éventuellement dans des places (◉) et peuvent se déplacer en franchissant des transitions. On a donc un réseau de Pétri "marqué". Il est caractérisé par la position et le mouvement des marques selon les arcs orientés.

Le franchissement des transitions par les "marques" est conditionnel Les arcs inhibiteurs (O—o/) ajoutent les conditions de franchissement. Une transition est franchissable si chaque place d'entrée de cette transition contient au moins une marque, et si chaque place connectée par un arc inhibiteur à cette transition ne contient aucune marque. Après que le franchissement d'une transition a été réalisée, une marque est soustraite à chaque place d'entrée de cette transition, et une marque est ajoutée à chaque place de sortie de cette transition.

Notre réseau de Pétri marqué représente 2 processus système, 2 programmes et la distribution du processeur pour chacun (figure 4.7). Chaque processus système est représenté par une courbe épaisse, fermée ayant une marque appelée "la marque processus". La courbe du processus 0 est simple, par contre, la courbe du processus 1 est plus complexe pour représenter l'introduction d'un nouveau processus (fonction la plus importante du processus 1). Chaque programme est représenté par une courbe épaisse mi-fermée mi-ouverte ayant éventuellement une marque venant de l'extérieur à un moment choisi par l'opérateur. Le fonctionnement du processeur est représenté par une courbe complexe, mince, ayant une marque appelée "la marque processeur".

Les places représentent les états du processeur et des processus :

. Pi_1 : l'état d'attente d'éxécution du processus i (i = 0,1,2,3)
. Pi_2 : l'état en cours d'éxécution du processus i, en même temps, l'occupation du processeur par ce processus.
. Pi_3 : l'existence d'un programme dans la position i (i = 2,3) de la mémoire.
. Pi_4 : le programme i éliminé de la mémoire
. Qi : l'état prêt du processeur pour le processus i
. E : l'existence d'une demande de chargement
. R : une demande de chargement a été refusée

Les transitions représentent les évènements qui entraînent un changement d'état :

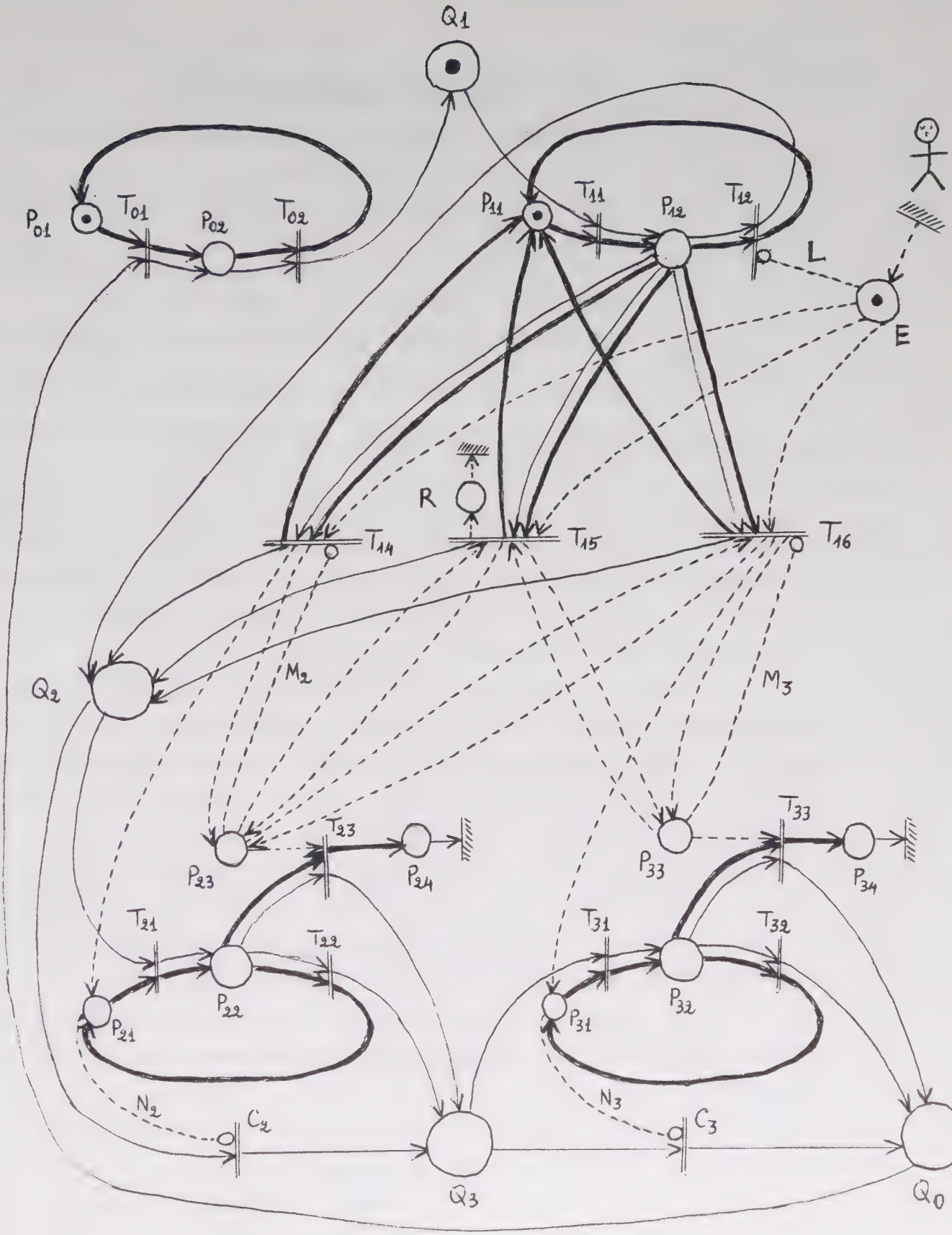


Figure 4.7.- Modèle de la gestion des processus du calculateur

Ti_1 : allouer le processeur pour le processus i

Ti_2 : le processus i abandonne provisoirement le processeur à cause d'une opération d'entrée-sortie

Ti_3 : le programme i abandonne définitivement le processeur par sa fin (i = 2,3).

Ci : l'état du processeur prêt pour le processus i devient l'état prêt pour le processus i + 1 (modulo 4, i = 2,3)

T_{14} : admettre le chargement d'un programme dans la position 2

T_{16} : admettre le chargement d'un programme dans la position 3

T_{15} : refuser le chargement par l'existence actuelle de 2 programmes dans la mémoire

Alors, avec ce réseau de Pétri marqué, on voit bien que la marque processeur se déplace sur la courbe mince, quant aux marques processus, elles se déplacent sur les courbes épaisses; et pendant le temps où un processus se trouve dans l'état "en cours d'éxécution", la marque processeur et la marque de ce processus se déplacent ensemble selon un chemin commun.

L'arc inhibiteur L oblige le processus 1 à examiner les conditions de chargement lorsqu'il y a une demande de chargement d'un programme. L'arc inhibiteur Mi (i = 2,3) empêche le chargement d'un nouveau programme dans la position i où existe déjà un programme. Les arcs orientés aller-retour $P_{23} - T_{16} - P_{23}$ assurent le choix de position dans la mémoire pour un nouveau programme en faveur de la position 2. Les arcs orientés aller-retour $P_{23} - T_{15} - P_{23}$ et $P_{33} - T_{15} - P_{33}$ conditionnent le refus de chargement.

Les arcs inhibiteurs N_i s'assurent qu'un programme existant aura le contrôle dans chaque cycle de distribution du processeur.

En conclusion, la modélisation du système par un réseau de Pétri marqué permet donc d'avoir une vue générale et dynamique de la gestion des processus concurrents du système.

C H A P I T R E V

GESTION DES FICHIERS SUR DISQUE

L'existence de programme s'éxécutant simultanément ne permet pas l'utilisation d'adresses absolues sur disque dans les opérations d'entrée-sortie. Ceci oblige à organiser les informations sur disque (des programmes et des données) en fichiers de sorte que toutes les manipulations sur ces informations par des programmes d'application sont réalisées à travers la gestion des fichiers.

1 - ORGANISATION DES FICHIERS -

L'installation VARIAN n'a qu'un seul disque ayant 2 faces. Chaque face a 203 pistes, chaque piste est fractionnée physiquement en 24 secteurs, chaque secteur contient les informations équivalentes à celles de 120 mots de 16 bits. Au total, la capacité du disque est d'un peu plus d'un Mégamot.

Le positionnement de la tête de lecture ou d'écriture sur disque n'a lieu qu'au début d'un secteur. Aussi, une lecture ou écriture sur disque est-elle obligée de commencer au début d'un secteur. De plus, une écriture sur disque se termine toujours à la fin d'un secteur. Autrement dit, l'unité adressable du disque est le secteur.

En tenant compte des caractéristiques précédentes du disque, le disque a été organisé en 2 régions principales : celle correspondant au répertoire des fichiers existants sur le disque et celle des fichiers proprement dit.

Chaque fichier occupe un espace continu sur le disque. Cet espace commence au début d'un secteur.

Le répertoire est constitué de blocs de longueur fixe. Chaque bloc appelé "descripteur" de fichier contient les renseignements nécessaires pour toutes les manipulations sur un fichier, tels que : le nom du fichier, l'adresse absolue et la longueur de l'espace réservé au fichier, la longueur de la

partie réellement écrite, et pour un fichier de module chargeable, on trouve dans son descripteur la longueur du corps du programme, la longueur du descripteur des adresses translatables et la liste des unités périphériques non-partageables nécessaires pour son éxécution.

2 - CREATION ET SUPPRESSION D'UN FICHIER -

Le répertoire est construit de telle manière que l'ordre des descripteurs soit le même que celui des fichiers auxquels ils se rapportent.

La création d'un fichier sur disque avec un nom et une longueur donnée est en fait la recherche d'un espace ayant cette longueur dans la région des fichiers, et ensuite, la création de son descripteur. Celui-ci contient le nom donné, le pointeur vers le début de l'espace réservé et la longueur de cet espace. Au moment de la création d'un fichier, la longueur de la partie réellement écrite sur ce fichier est évidemment égale à 0. La configuration du disque après des créations consécutives de fichiers est montrée dans la figure 5.1(a).

La suppression d'un fichier sur disque est en fait la libération de l'espace réservé à ce fichier et la libération de la place occupée dans le répertoire par son descripteur. La suppression des fichiers produit des espaces libérés en conservant la taille de chacun (figure 5.1(b)). La réorganisation du répertoire et des espaces sera faite au moment de la compression du disque.

Quand il y a des espaces libérés, la recherche d'un espace pour un nouveau fichier est réalisée tout d'abord sur les espaces libérés, et s'il existe des espaces libérés satisfaisant la longueur de l'espace demandé, l'espace libéré ayant la longueur la plus proche de celle demandée sera sélectionné. Sinon, l'espace pour le nouveau fichier se trouvera dans la partie libre de la région des fichiers (figure 5.1 (c)).

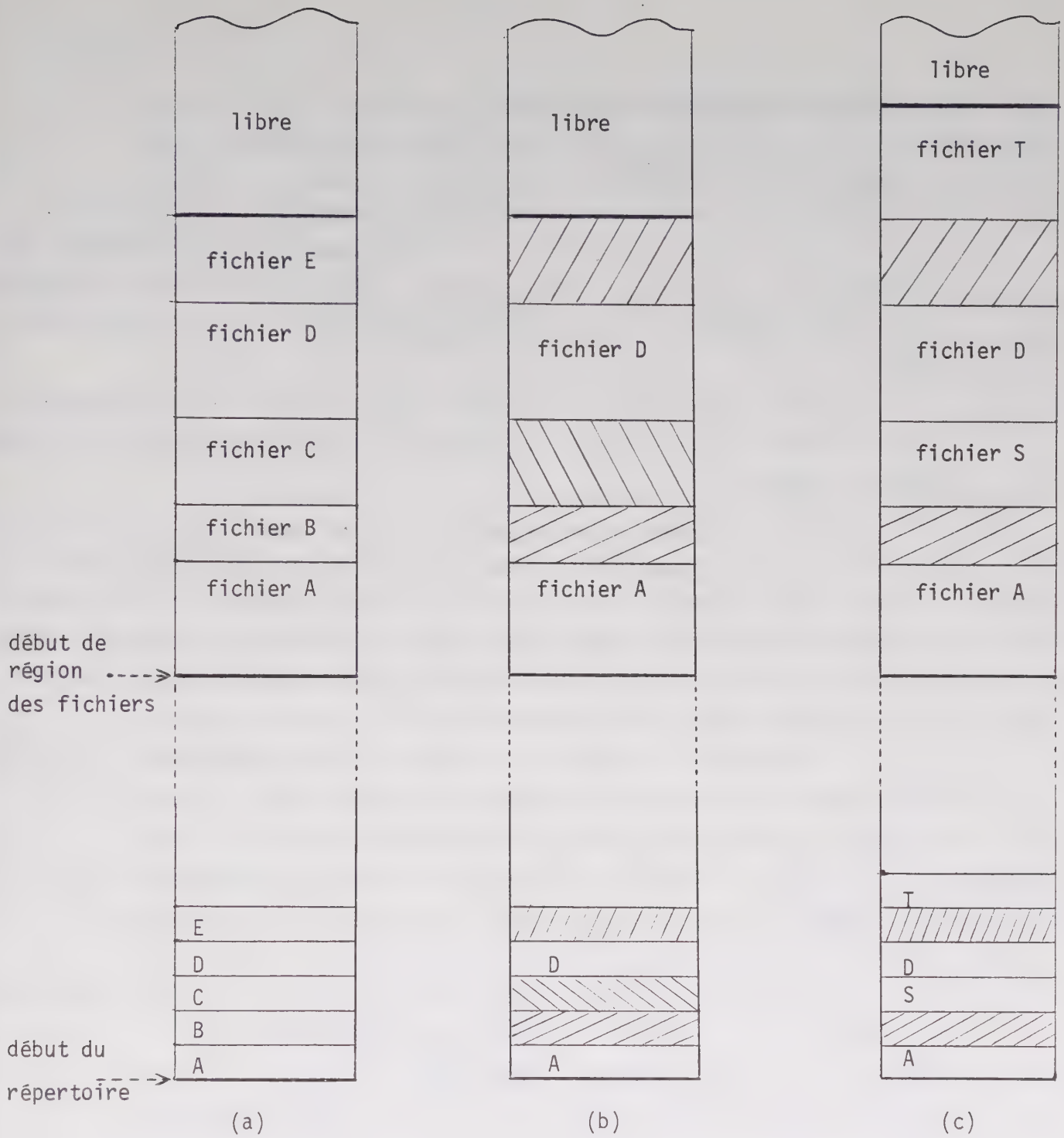


Figure 5.1. Configuration du disque après les créations et suppressions des fichiers

3 - ACCES AUX INFORMATIONS SUR DISQUE -

3.1. Contrôle d'accès

Le disque est une unité périphérique partageable. Un fichier sur disque est une ressource partageable en lecture, mais c'est une ressource non-partageable en écriture. En tenant compte de cette caractéristique, on applique une méthode dynamique pour la distribution des fichiers aux programmes coexistants.

Selon cette méthode, un fichier ne sera attaché à un programme qu'à partir du moment où celui-ci l'aura ouvert. Il n'appartiendra plus au programme après sa fermeture.

Parmi les renseignements contenus dans le descripteur, on s'intéresse à présent aux renseignements sur l'état du fichier qui permettent de savoir à un moment donné le possesseur du fichier et l'action demandée sur ce fichier. Un fichier peut avoir l'un des états suivants : libre (sans possesseur), occupé par un programme (en lecture seulement ou en écriture et lecture) ou occupé par 2 programmes (en lecture seulement).

Une demande d'ouverture d'un fichier est donc une demande d'occupation de ce fichier. Cette demande est admise dans les deux cas suivants :

- ou bien le fichier est libre,
- ou bien le fichier est occupé en lecture seulement par un programme, et la nouvelle demande est aussi une demande pour la lecture seulement.

L'ouverture d'un fichier est en fait la vérification de l'état actuel du fichier, et si la demande d'occupation est admise, d'une part, l'état du fichier dans le descripteur sur disque sera modifié, et d'autre part, le descripteur sera chargé en mémoire pour servir à gérer ultérieurement les opérations d'entrée-sortie sur ce fichier.

De même, la fermeture d'un fichier, du point de vue de la gestion des ressources, correspond à l'abandon de ce fichier. Le traitement correspond à examiner l'état actuel du fichier, retirer un possesseur et modifier de nouveau le descripteur sur disque.

Autrement dit, l'ouverture ou la fermeture d'un fichier entraîne une lecture et une écriture sur le répertoire du disque en fonction de l'examen de

l'état du fichier. Comme on l'a vu, un programme peut donner le contrôle à un autre programme pendant une opération d'entrée-sortie. Alors, probablement il y aura un moment où les deux programmes réaliseront simultanément l'ouverture et/ou la fermeture d'un même fichier, et par conséquent, l'état du fichier deviendra indéterminable.

Pour éviter ce phénomène, le répertoire du disque est considéré comme une ressource non-partageable, même en cas de lecture. Une variable sert donc à indiquer l'état libre ou occupé du répertoire. L'ouverture ou la fermeture d'un fichier doit tester le contenu de cette variable avant de le modifier pour indiquer l'occupation du répertoire (figure 5.2).

En outre, la distribution dynamique des fichiers à la demande des programmes peut entraîner le phénomène d'interblocage. Exemple : les deux programmes (numéros 2 et 3) coexistent, chacun demande l'ouverture de 2 fichiers A et B en écriture, mais dans les ordres différents : le programme 2 demande A puis B, pendant que le programme 3 demande B puis A. On peut avoir un moment où le programme numéro 2 attend indéfiniment le fichier B occupé par le programme 3, en même temps, le programme 3 attend indéfiniment le fichier A occupé par le programme 2 (figure 5.3).

Pour éviter le phénomène d'interblocage, on applique la solution suivante : quand une demande d'occupation d'un fichier n'est pas satisfaite, le programme demandeur sera bloqué sans donner le contrôle à "l'ordonnateur", et ensuite, il sera éliminé hors de la mémoire par l'intervention de l'horloge. Cette terminaison forcée permet de libérer les fichiers occupés pour ce programme.

3.2. Méthodes d'accès

L'accès aux informations d'un fichier est donc réalisé tout d'abord par le sous-programme d'ouverture du fichier. Si cette ouverture est admise, le descripteur, après avoir été modifié en ce qui concerne l'état du fichier, est chargé en mémoire, et est alors, appelé "le bloc de contrôle" du fichier.

Un pointeur dynamique de lecture et un pointeur dynamique d'écriture sont créés dans le bloc de contrôle. Au moment de l'ouverture du fichier, ces pointeurs pointent tous les deux vers l'adresse absolue de début du fichier.

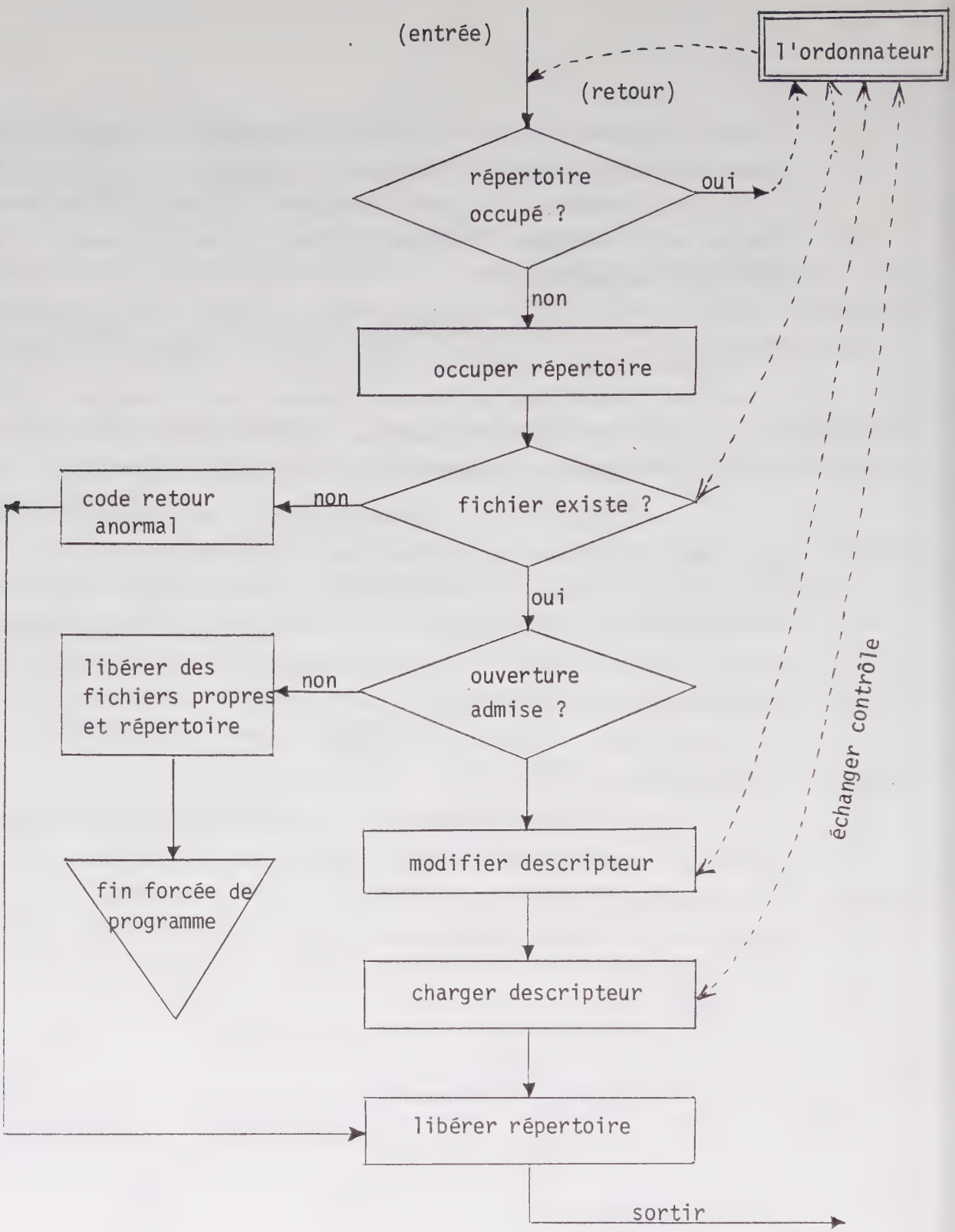


Figure 5.2.- Sous-programme d'ouverture d'un fichier

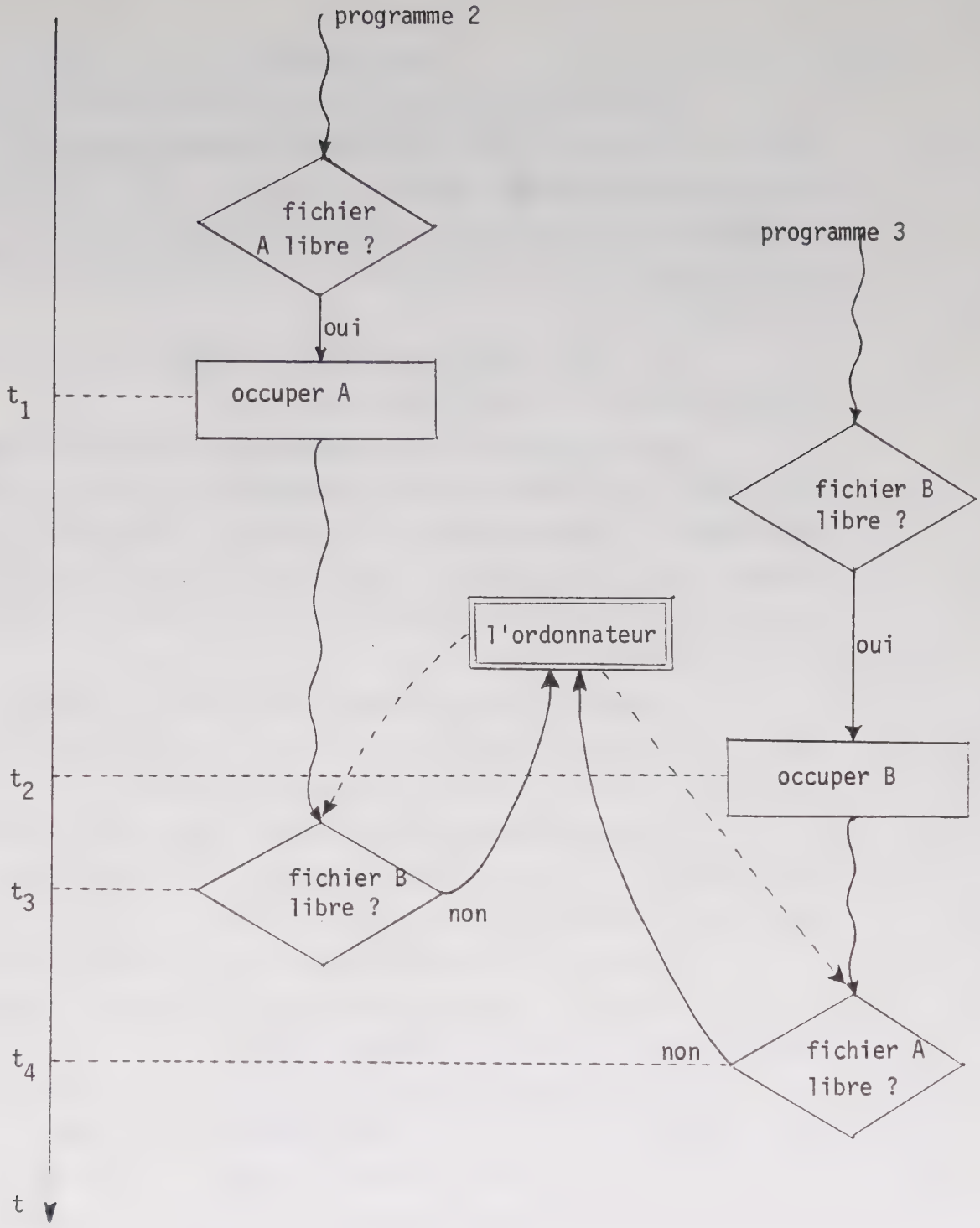


Figure 5.3.

L'éxécution des programmes 2 et 3 avec l'écoulement de temps

Chaque demande d'échange d'informations (lecture ou écriture) entre un fichier et la mémoire centrale appelle un sous-programme d'entrée-sortie sur disque. Ce sous-programme, résidant en mémoire dans la zone du système, demande l'adresse absolue sur disque comme un paramètre explicite de l'échange (se référer au chapitre "Gestion des opérations d'entrée-sortie par des interruptions élémentaires").

Il y a deux méthodes d'accès aux informations d'un fichier : méthode d'accès séquentielle et méthode d'accès directe.

Avec la méthode d'accès séquentielle, le sous-programme d'entrée-sortie sur disque appelé réalise un échange en utilisant un pointeur dynamique pour trouver l'adresse absolue de la fraction d'informations concernant cet échange. Après chaque échange, le pointeur dynamique est augmenté d'une valeur en fonction de la longueur de dernier échange pour préparer l'échange suivant (figure 5.4 (a)).

Au moment de la fermeture d'un fichier, le pointeur dynamique d'écriture qui ne pointe plus vers l'adresse absolue du début du fichier permet de connaître la fin de la partie réellement écrite du fichier.

Le dépassement de l'espace réservé pour le fichier (en cas d'écriture) ou l'arrivée à la fin de sa partie réellement écrite (en cas de lecture) sera détectée par la méthode d'accès séquentielle.

La méthode d'accès séquentielle permet d'écrire les informations d'un fichier et de les relire ou réécrire ultérieurement totalement ou partiellement à partir du début du fichier.

Avec la méthode d'accès directe, il faut savoir l'adresse relative de la fraction d'informations choisie par rapport au début de fichier. Cette adresse relative est fournie comme un paramètre de chaque écriture ou lecture utilisant la méthode d'accès directe. Alors, le sous-programme d'entrée-sortie sur disque appelé se servira de ce paramètre et du bloc de contrôle en mémoire pour calculer l'adresse absolue de la fraction d'informations choisie (figure 5.4 (b)).

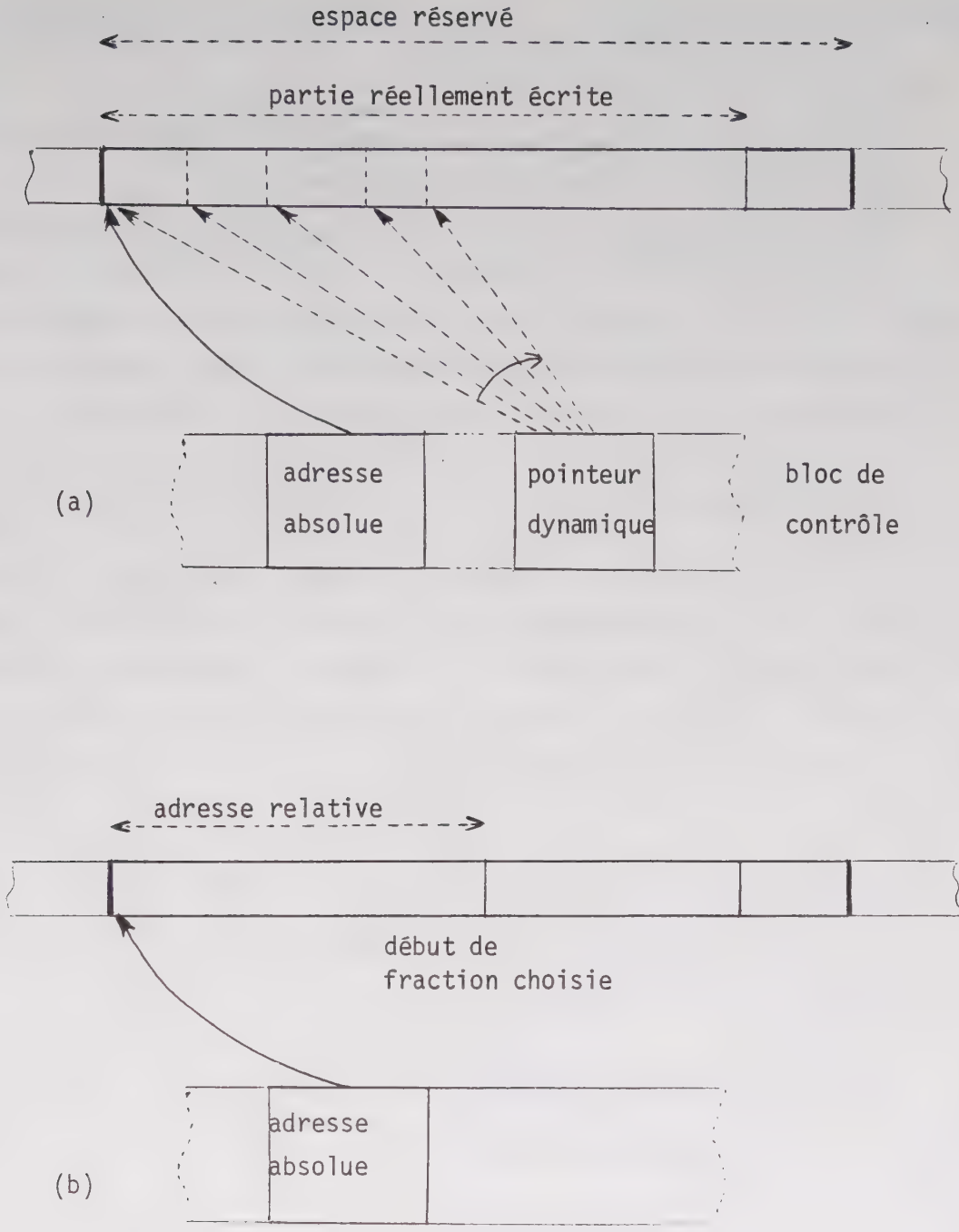


Figure 5.4 (a) Méthode d'accès séquentielle
(b) Méthode d'accès directe

Le dépassement de la fin de l'espace réservé au fichier sera détecté par la méthode d'accès directe.

La méthode d'accès directe permet de lire ou modifier une fraction d'informations choisie à volonté sur un fichier.

Les pointeurs dynamiques de la méthode d'accès séquentielle sont sauvegardés dans des positions différentes du bloc de contrôle, tandis que l'adresse relative pour la méthode d'accès directe est fournie à chaque lecture ou écriture. Ceci permet de réaliser simultanément sur un fichier ouvert les lectures et les écritures avec les méthodes d'accès séquentielle et directe.

Le bloc de contrôle d'un fichier se trouve dans la zone mémoire du programme concerné. La modification des pointeurs dynamiques dans ce bloc est à la charge d'un sous-programme gérant la méthode d'accès séquentielle. Le calcul de l'adresse absolue de la fraction d'informations choisie à volonté est à la charge d'un sous-programme gérant la méthode d'accès directe. Ces deux sous-programmes se trouvent dans la zone mémoire du programme concerné, s'occupent du niveau logique de l'échange d'informations. Le déclenchement de l'échange d'informations au niveau physique est réalisé par un sous-programme d'entrée-sortie qui est résidant dans la zone mémoire du système. La gestion de l'échange au niveau physique est à la charge de la partie de traitement d'interruption du disque, résidant dans la zone mémoire du système.

La figure 5.5. nous donne l'aspect global de l'échange d'informations entre la mémoire centrale et le disque.

4 - MANIPULATION SUR L'ENSEMBLE DES FICHIERS

* Sauvegarde du disque

Les fichiers stockés sur disque du VARIAN peuvent tous être sauvegardés sur une bande magnétique, et à un moment ultérieur, s'il est nécessaire, ils seront rechargés sur disque, fichier par fichier au choix. Ceci permet de conserver en même temps plusieurs versions différentes d'un même fichier sans occuper des espaces sur disque.

Il est également possible de lire le contenu du répertoire sur disque et de lister les fichiers sur une bande de sauvegarde.

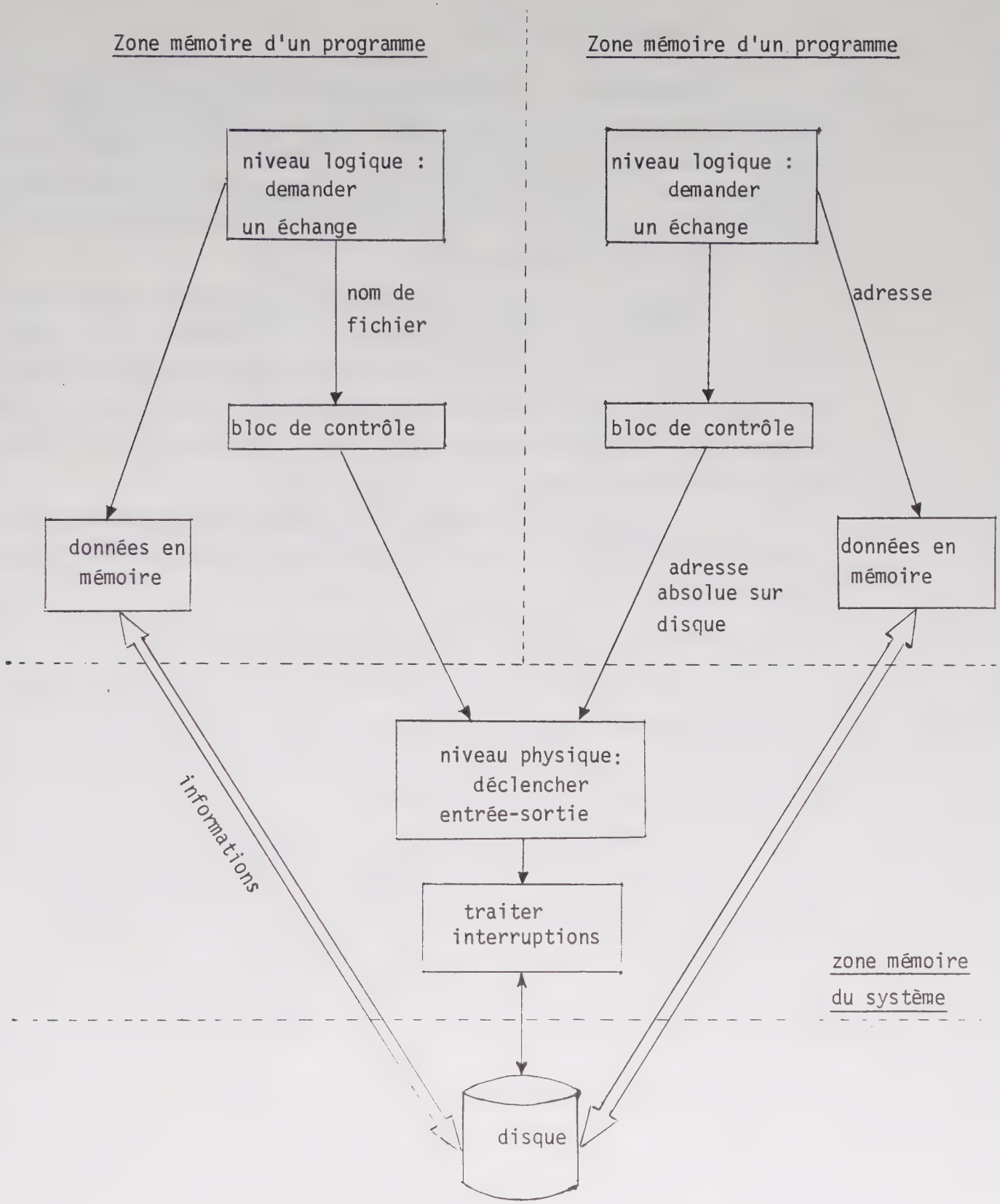


Figure 5.5.

Niveau logique et niveau physique de l'échange d'informations entre la mémoire et le disque.

* Compression du disque

La création et la suppression des fichiers provenant dans un ordre arbitraire produisent des espaces libérés dans la région des fichiers et des places libérées dans le répertoire (voir la figure 5.1 (c)). Avec la politique de la recherche d'un espace pour un nouveau fichier (voir le paragraphe 2 de ce chapitre), il se peut qu'un nouveau fichier soit créé dans la partie libre bien que des espaces libérés existent.

Evidemment, le nombre de places dans le répertoire est limité, de même que la taille de la région des fichiers. En conséquence, il y aura probablement un moment où un nouveau fichier devra être créé dans la partie libre, alors que ou bien la partie libre actuelle est trop petite, ou bien le répertoire est complet (y compris des places libérées). C'est le phénomène de la saturation du disque.

On réalise donc la compression du disque pour récupérer des espaces libérés dans la région des fichiers et des places libérées dans le répertoire (figure 5.6.).

* Traitement de fin d'utilisation des fichiers

L'ouverture d'un fichier entraîne une modification de son descripteur sur disque pour montrer que le fichier est occupé. Naturellement, la fermeture de ce fichier est exigée plus tard pour le libérer de son possesseur, autrement dit, pour modifier à nouveau son descripteur.

Si la fermeture de fichier n'est pas lancée par le programme utilisateur, le système dispose d'une procédure qui libère tous les fichiers encore occupés par un programme après la fin d'éxécution de celui-ci.

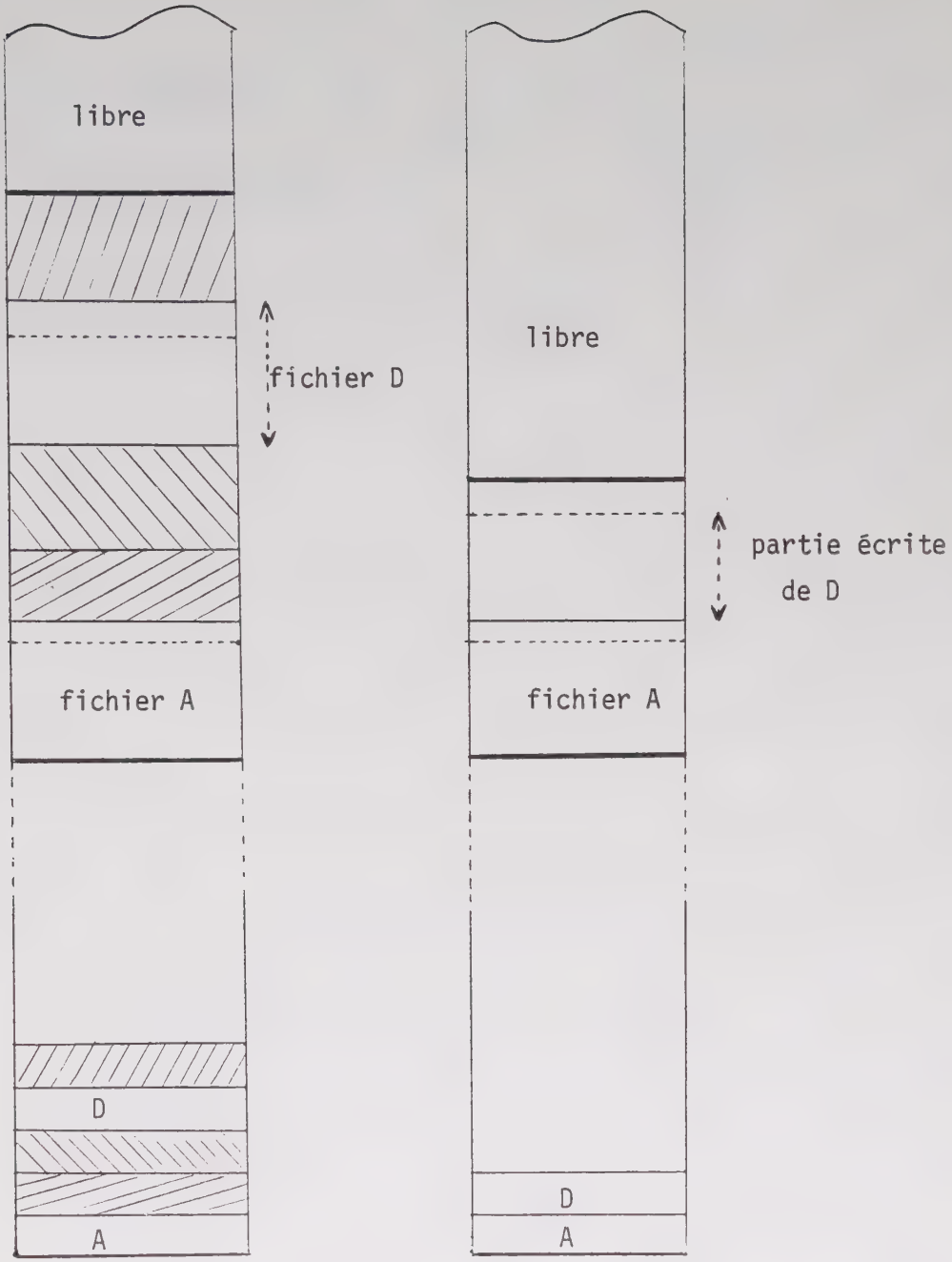


Figure 5.6. Compression du disque

a) Avant la compression

b) Après la compression

C H A P I T R E VI

FACILITES D'EXPLOITATION DU CALCULATEUR

Dans les chapitres précédents, on a exposé les solutions fondamentales pour réaliser le but du projet : le fonctionnement parallèle entre le processeur et des unités périphériques, et l'éxécution simultanée de 2 programmes coexistants en mémoire plus 2 processus du système. Ces solutions constituent les composants principaux d'un système de base du calculateur.

En outre, on a ajouté des procédures visant à faciliter l'exploitation quotidienne du calculateur et la mise au point des programmes d'application, en tenant compte des caractéristiques du matériel et des applications du Service de Microdensitométrie.

Tout cela permet de mettre ce système en service.

1 - FACILITES DE L'EXPLOITATION QUOTIDIENNE

1.1. Utilisation efficace de la bande de sauvegarde

La bande de sauvegarde du disque contient le système et tous les fichiers existants sur le disque.

En tenant compte de la caractéristique physique du disque qui oblige à commencer la lecture et l'écriture sur disque chaque fois en début d'un secteur, les informations sur la bande de sauvegarde sont organisées de telle manière que l'échange d'informations entre bande et disque soit avantageux. Pour cela, sur une bande de sauvegarde, les informations d'un fichier (y compris celles du système) sont regroupées en enregistrements de longueur fixe qui est un multiple de la longueur du secteur, sauf pour le dernier enregistrement de chaque fichier.

Cette structure permet d'utiliser une bande de sauvegarde pour le chargement d'un programme en mémoire.

De même, grâce à une suite d'instructions appelée 'bootstrap", on peut charger le système en mémoire aussi bien à partir d'une bande de sauvegarde qu'à partir du disque. Ceci permet au calculateur de continuer à fonctionne sans disque avec les programmes qui n'ont pas besoin de disque.

1.2. Ensemble de messages d'observation et de diagnostic

- En cas de besoins,l'opérateur peut observer l'exploitation du calculateur à l'aide d'une liste des programmes existants en mémoire. Cette liste permet de savoir le nom et le numéro des programmes.

- Les informations sorties sur le papier du télétype central constituent le journal d'exploitation. Elles permettent de connaître le démarrage du système, le commencement et la fin d'éxécution de chaque programme (fin normale, fin due à une intervention de l'opérateur, fin due à une intervention de l'horloge).

- Au moment du chargement d'un programme en mémoire, les messages de diagnostic permettent à l'opérateur de savoir s'il y a :

.indisponibilité physique d'une unité périphérique non-partageable nécessair
.anomalie du disque ou de la bande,
. insuffisance de mémoire,
.nécessité d'une bande à écrire,
.absence du programme à charger.

1.3. Traitement des cas anormaux

- Comme on l'a vu dans le chapitre 3, pendant une opération d'entrée-sortie sur disque ou bande, lorsqu'une erreur matérielle est reconnue, l'opération d'entrée-sortie en question sera relancée par le système plusieurs fois dans le but de récupérer si possible cette erreur. En ce qui concerne les erreurs humaines de frappe sur le clavier des télétypes, le système permet à l'opérateur de reintroduire tout le message d'entrée.

- De plus, la lecteure sur une imprimante donne la possibilité de recommencer à éxécuter le programme concerné à partir d'un point choisi d'avance par le programmeur, lorsque l'opérateur introduit un caractère spécial ("CONTROL + X"), par exemple, pour imprimer de nouveau le dernier message de

demande de données. Ceci facilite la conversation entre l'opérateur et le calculateur qui est une caractéristique remarquable des programmes d'application.

Pour cela, le sous-programme de lecture sur imprimante à clavier demande parmi les paramètres une adresse dans le programme principal. Au moment de déclenchement de la lecture, cette adresse et le numéro du programme sont sauvegardés. Pendant la lecture au clavier, lorsque le caractère spécial est reconnu, la partie de traitement consulte la variable "ordre" afin de connaître le processus interrompu à cause de cette interruption. Si c'est bien le programme qui a déclenché la lecture, l'adresse de retour après le traitement de cette interruption sera remplacée par l'adresse sauvegardée. Sinon, l'adresse sauvegardée sera mise dans le bloc d'état du programme concerné.

- Une imprimante a aussi la possibilité de lancer des sonneries en attendant la réponse de l'opérateur. C'est le résultat de l'association entre le sous-programme de lecture, la partie de traitement d'interruption de lecture et le traitement d'interruption de l'horloge à travers un compteur commun (figure 6.1).

- La disponibilité des unités périphériques non-partageables est vérifiée avant le chargement d'un programme. Cependant une unité périphérique pourra devenir indisponible après un arrêt provisoire pour le changement de volume d'information (par exemple : changement de bandes sur le dérouleur de bande, changement de films sur le microdensitomètre).

L'indisponibilité d'une unité périphérique en cours de fonctionnement ne provoque pas une interruption (sauf le cas de la coupure de la ligne de télétransmission). Aussi, avant de déclencher une opération d'entrée-sortie, la phase 1 vérifie à nouveau l'état de l'unité concernée. Si cette unité n'est pas prête, la phase 1 informe cette situation à l'opérateur, puis elle se met en attente.

- Dans le cas où une unité périphérique concernant l'éxécution d'un programme tombe en panne, ce programme sera bloqué. L'opérateur peut se servir d'une commande du processus 1 pour supprimer l'éxécution de ce programme afin de le recharger ou charger un autre programme.

Dans le cas où l'éxécution d'une commande du processus 1 est bloquée, l'opérateur peut utiliser le processus 0 pour sortir de cette commande afin de lancer une autre commande.

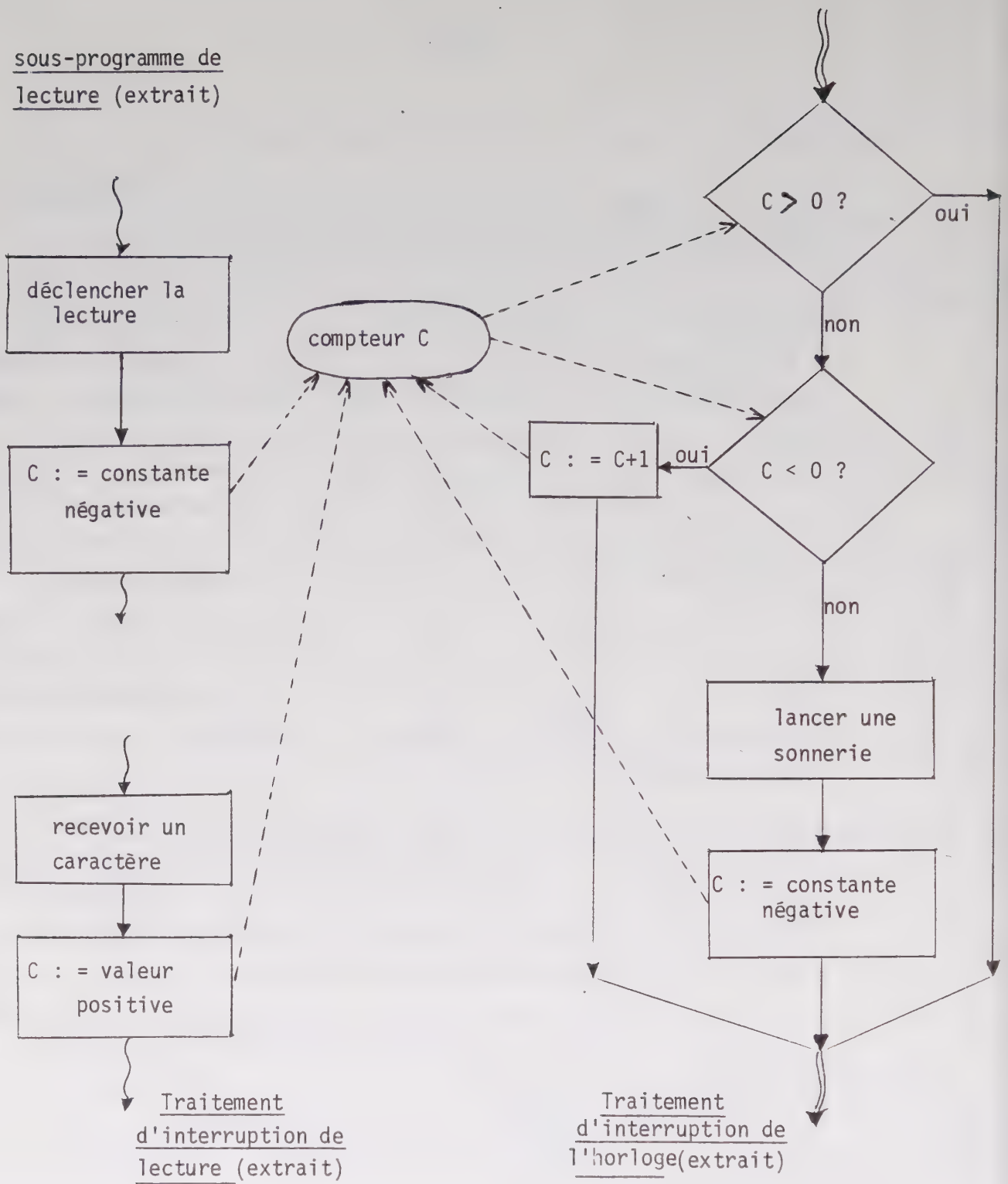


Figure 6.1.- Le lancement des sonneries en attendant la lecture sur une imprimante à clavier

1.4. Intervention directe sur les unités périphériques

Le processus système numéro 1 ne dispose que des unités périphériques partageables, comme on l'a vu dans le chapitre 4. Malgré cela, on permet au processus 1 d'accéder au dérouleur de bande et au microdensitomètre lorsqu'une intervention directe de l'opérateur est nécessaire.

- Le montage manuel d'une bande sur le dérouleur de bande n'assure pas toujours l'état normal du dérouleur de bande. Il s'agit là d'un défaut électronique difficile à pallier. Pour éviter des erreurs d'écriture au début de la bande, le rebobinage est recommandé tout de suite après le montage manuel. Le processus 1 est capable de le faire par une commande de l'opérateur.

- L'éxécution d'un programme peut s'arrêter anormalement pour une cause ou une autre. Pour conserver les résultats écrits sur une bande, l'opérateur a la possibilité de marquer une fin de fichier sur la position actuelle de la bande dans le dérouleur.

- Au moment du chargement du système en mémoire, le système considère que la position du chariot du microdensitomètre est à 0 (position initiale). Après quoi, tous les déplacements du chariot pendant l'éxécution des programmes d'application sont mémorisés par le système.

La mise manuelle du chariot à la position 0 demande un réglage délicat. Or, chaque programme ne remet pas systématiquement le chariot à la position 0 après la fin de son éxécution. Dans le cas d'une exploitation discontinue (par exemple, le calculateur s'arrête pendant la nuit), pour éviter de régler la position du chariot à chaque fois que l'on charge le système, on permet au processus 1 de remettre le chariot du microdensitomètre à la position 0 avant d'arrêter le calculateur.

2 - FACILITES DE LA MISE AU POINT D'UN PROGRAMME

- Comme on l'a vu dans le chapitre 2, toutes les phases de la préparation d'un programme sont réalisées sur l'ordinateur UNIVAC. Grâce à un programme utilitaire, une imprimante à clavier du VARIAN peut jouer le rôle d'un terminal connecté à l'UNIVAC, il peut donc participer à la préparation d'un module chargeable.

Un module chargeable peut être stocké sur un disque de l'UNIVAC. Ensuite, via une ligne de transmission, ce module est transmis vers le disque du VARIAN afin d'être chargé en mémoire pour s'éxécuter. Sur le VARIAN, la préparation d'un nouveau programme peut alors se réaliser parallèlement à l'éxécution d'un autre programme.

- Le changement de nom d'un fichier existant est possible. Ceci permet d'avoir au cours de la mise au point d'un programme plusieurs versions différentes d'un même programme sur le disque et ainsi, de faciliter le test de l'éxécution du programme.

- Les sous-programmes permettant de sortir sur une imprimante le contenu momentané des registres et des zones intéressantes en mémoire peuvent être insérés dans un programme, aux endroits choisis par le programmeur. Ceci facilite la recherche des erreurs de programmation. De plus, ces sous-programmes fonctionnent en relation avec les clés logiques. Alors, en modifiant les clés logiques, on peut ne sortir le contenu de la mémoire qu'en cas de besoin.

- Un programme utilitaire permet de sortir sur une imprimante le contenu d'une bande magnétique, en gros ou en détail en fonction des clés logiques. Ceci a pour but d'aider à tester l'exactitude des programmes utilisant une bande.

3 - EVALUATION DU SYSTEME

Le système de multiprogrammation simple (deux programmes au maximum) du calculateur VARIAN 620/L-100 est mis en service. L'exploitation réelle nous permet d'évaluer ce nouveau système.

3.1. Fiabilité

Le fonctionnement du système est stable. Les résultats des programmes d'application sous le nouveau système sont les mêmes que sous le système de monoprogrammation.

Toutefois, à cause de la faiblesse de la structure physique du calculateur, il reste les 2 défauts suivants :

- La protection des zones en mémoire du système et de chaque programme n'est pas assurée par le contrôle du système. Toute la mémoire est accessible

en lecture ou en écriture.

- Les codes opération erronés des instructions ne sont pas traités par le système. Dans ce cas, le fonctionnement du calculateur s'arrête. Il faut alors recharger le système.

3.2. Efficacité

Le but du projet est atteint. Le fonctionnement parallèle entre le processeur et les unités périphériques, et l'éxécution simultanée de 2 programmes permettent de profiter du temps d'attente des opérations d'entrée-sortie, notamment les entrées via les claviers qui ont besoin de l'intervention humaine. Il est donc possible d'avancer en même temps deux travaux différents et de gagner beaucoup de temps au total.

Le système est assez souple pour s'adapter au besoin d'exploitation originale et à la manière conversationnelle des programmes d'application, et ainsi de profiter au mieux des caractéristiques propres à l'installation VARIAN 620/L-100 du Service de Microdensitométrie.

Enfin on peut maintenant ""tuer"" un programme sans avoir à arrêter le calculateur manuellement, donc sans recharger le système.

3.3. Modifiabilité

La préparation d'un "module système"se réalise sur l'ordinateur UNIVAC, selon le principe de préparation utilisé pour un module programme. Par conséquent, ce module peut être transmis sur la ligne à partir de l'UNIVAC vers le disque du VARIAN, et on peut sauvegarder plusieurs versions de "module système" sur le disque du VARIAN et puis en choisir un pour l'exploitation. Cette méthode facilite et accélère la modification et la mise au point d'un système.

Le système contient des symboles de référence (par exemple, le nom des sous-programmes résidants) utilisés par les programmes. Ces symboles sont mis dans des adresses précises de la mémoire pour que les modules chargeables restent valables dans le cas général de modifications apportées au système qui ne changent ni les noms, ni les adresses absolues des symboles (Fig.6.2).

Le système a une structure modulaire afin de faciliter la modification ultérieure, particulièrement, le développement des unités périphériques.

Il y a encore 2 lignes d'interruptions réservées aux nouvelles unités périphériques. De plus, l'interface asynchrone multiple peut servir au maximum 8 unités périphériques, et les lignes de transmission permettent de

connecter le calculateur VARIAN avec les ordinateurs puissants du Centre inter-régional de calcul électronique (CIRCE). Dans ce dernier cas le logiciel reste à mettre au point car les protocoles conversationneles sont différents.

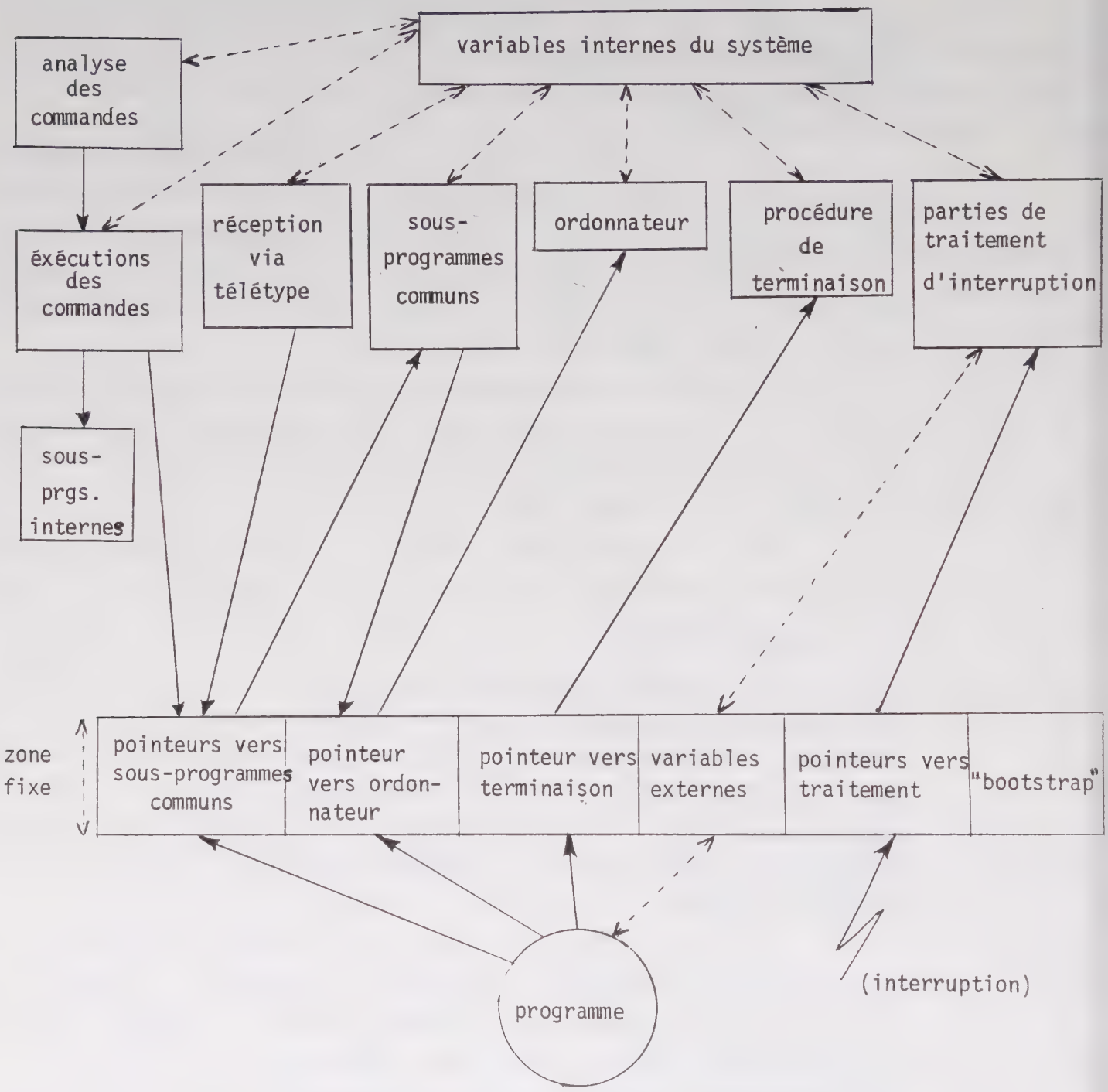


Figure 6.2.- La communication entre un prg. et les composants du système

A N N E X E 1

GUIDE DE PROGRAMMATION

Le nouveau système du calculateur VARIAN permet à 2 programmes d'application de s'éxécuter simultanément. Ceci nécessite des modifications dans la programmation par rapport à l'ancien système de monoprogrammation.

Cette annexe présentera ces modifications. Tous les noms des symboles et des sous-programmes sont conventionnels.

1 - MODIFICATIONS DE STRUCTURE D'UN PROGRAMME

1.1. Liste des unités périphériques non-partageables nécessaires

Les unités périphériques non-partageables de l'installation VARIAN sont le dérouleur de bande, le microdensitomètre, les imprimantes KSR43 et Decwriter 3 et les modems de 300 bauds et 1200 bauds.

Un programme est obligé de déclarer la liste des unités périphériques non-partageables nécessaires pour son éxécution. Cette déclaration doit être faite par une instruction de la forme :

D ECLARE$* EQU <n>

avec <n> : une valeur entière.

Cette valeur est en réalité une suite de bits dont chacun correspond à une unité périphérique selon la convention suivante :

. Bit numéro 0 : imprimante (KSR43 ou Decwriter 3)
. Bit numéro 1 : modem (300 bauds ou 1200 bauds)
. Bit numéro 2 : microdensitomètre
. Bit numéro 3 : dérouleur de bande
. Bit numéro 4 : bande à écrire

Les bits sont numérotés de droite à gauche. Le bit ayant la valeur 1

indique le besoin de l'unité périphérique correspondante.

Par exemple, l'instruction

```
DECLARE$*    EQU    035
```

informe au système que le programme a besoin d'une imprimante, du micro-densitomètre, du dérouleur de bande et d'une bande magnétique à écrire (figure A.

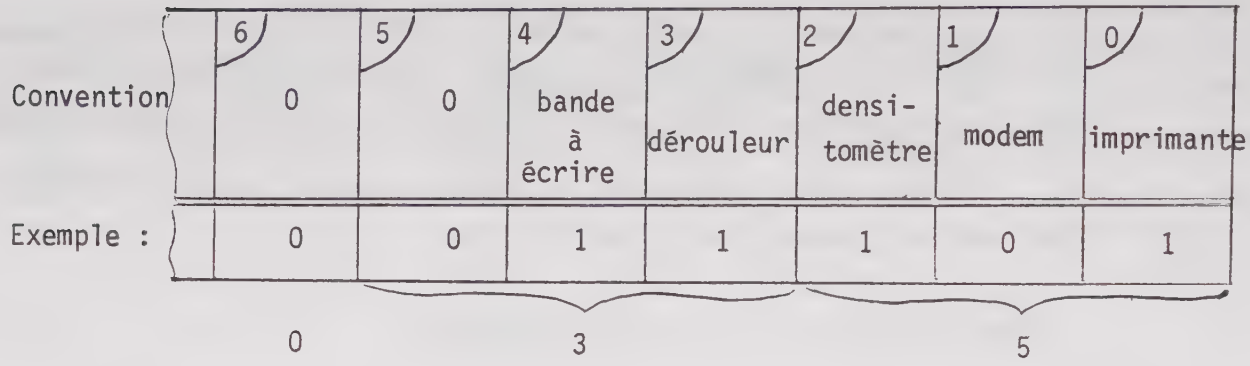


Figure A.1. Liste des unités périphériques non partageables nécessaires.

Même si le programme n'a besoin d'aucune unité périphérique non-partageable, cette déclaration est obligatoire, elle sera :

```
DECLARE$*    EQU    0
```

1.2. Procédure de terminaison

A la ou les sortie(s) prévue(s) au cours de l'éxécution du programme le programme redonne volontairement le contrôle au système par une instruction "saut" inconditionnel (JMP) ou conditionnel vers la procédure de terminaison normale appelée TERMINE$ qui se trouve en mémoire dans la zone du système.

1.3. Invalidité de quelques types d'instruction

Toutes les instructions concernant les clés physiques sur le panneau du calculateur ne sont plus acceptables (par exemple, les instructions de type JSSn avec n = 1, 2, 3). Elles sont remplacées par l'utilisation d'un sous-programme appelé KEY$ (voir le paragraphe 3.2. de cette annexe).

Hormis le cas d'unités périphériques non partageable et possédant en propre une ligne d'interruptions (dérouleur et densitomètre), l'utilisation des instructions de type entrée-sortie en dehors des sous-programmes

d'entrée-sortie du nouveau système est interdite. Par conséquent, toutes les procédures (macro-instructions) et toutes les sous-programmes de l'ancien système qui comporte directement ou indirectement des instructions de type entrée-sortie ne sont plus valables.

2 - SOUS PROGRAMMES D'ENTREE-SORTIE

Le changement fondamental de la gestion des opérations d'entrée-sortie demande une nouvelle série de sous-programmes d'entrée-sortie, mise à la disposition de l'utilisateur.

Tous les sous-programmes utilisent au moins le registre A. Certains demandent en plus le registre B et registre X.

2.1. Pour le microdensitomètre

2.1.1. Lecture à une position définie du chariot

CALL LIREPH$, nignore, nlire, abuffer, asignal

- nignore : nombre de caractères qui ne seront pas lus; ce nombre doit être pair
- nlire : nombre de caractères à lire; ce nombre doit être pair;
- abuffer : adresse du début de la zone mémoire recevant les informations lues
- asignal : adresse d'un mot où le signal de fin totale d'opération sera positionné.

2.1.2. Mouvement du chariot

CALL MOVEPH$,valposition, asignal

- valposition : position du chariot à atteindre après le mouvement, par rapport à la position 0, en unité élémentaire de 12,5 microns
- asignal : voir LIREPH$

2.1.3. Définition de la valeur "raster" physique

CALL GETRAST$

Ce sous programme utilise les registres A et B pour contenir ses résultats.

. Dans le registre A : valeur "raster" physique

. Dans le registre B : valeur "raster" en unité 25 microns

C'est un sous-programme d'entrée-sortie spécial qui s'éxécute sans utiliser le traitement d'interruption, car il compte le nombre de données lisibles dans la zone lisible du tambour du densitomètre. Il faut donc s'assurer que toutes les opérations d'entrée-sortie sur le microdensitomètre ont été terminées avant d'appeler ce sous-programme. Ceci est vérifiable si toutes les opérations sur le densitomètre utilisent un mot unique de signal de fin.

2.2. Pour la bande magnétique

2.2.1. Lecture d'un enregistrement

CALL LIREMT$, astart, vallg, aflag, asignal

- astart : adresse du début de la zone mémoire recevant les informations lues
- vallg : nombre de mots à lire
- aflag : adresse d'un mot contenant le code de retour de sous-programme.
- asignal : adresse d'un mot où le signal de la fin totale d'opération sera positionné.

* Signification du code de retour :

. Code = 0 : fin normale

. Code = 1 : erreur de parité

. Code = 2 : le nombre de mots à lire est différent de la longueur réelle de l'enregistrement lu.

. Code = 3 : fin de fichier.

En cas d'erreur de parité, il est inutile de rappeler ce sous-programme car il essaye de recommencer cette opération plusieurs fois avant de signaler l'erreur de parité.

Le nombre de reprises est enregistré dans la variable PARN$. La longueur réelle de l'enregistrement lu est mise dans la variable LGENRMT$.

2.2.2. Ecriture d'un enregistrement

CALL ECRIMT$, astart, vallg, aflag, asignal

- astart : adresse du début de la zone mémoire contenant les informations à écrire sur la bande
- vallg : nombre de mots à écrire

- aflag et asignal : voir LIREMT$

* Code de retour : 0 et 1 : voir LIREMT$

Le nombre de reprises est mis dans la variable PARN$.

2.2.3. Mouvement de la bande

CALL MOVEMT$, valtype, asignal

- valtype : type d'opération, sous la forme 0n00, dans laquelle :

. n = 1 : remise à l'état normal du dérouleur

. n = 4 : écriture d'une fin de fichier

. n = 5 : saut d'un enregistrement en avant

. n = 6 : saut d'un enregistrement en arrière

. n = 7 : rebobinage

2.3. POUR LE TETETYPE CENTRAL (ASR33)

2.3.1. Ecriture d'une chaîne de caractères

CALL ECRITY$, vallgc, astart, asignal

- vallgc : nombre de caractères à écrire

- astart : adresse du début de la zone mémoire contenant la chaîne de caractères à écrire.

- asignal : voir LIREMT$

2.3.2. Attente d'une chaîne de caractères d'entrée via le télétype central

CALL WAITREP$, vallg, astart

- vallg : nombre de mots réservés pour les informations entrées

- astart : adresse du début de la zone mémoire réservée pour les informations entrées

2.4. POUR UNE IMPRIMANTE A CLAVIER (KSR43 ou DECWRITER 3)

2.4.1. Ecriture d'une chaîne de caractères

CALL ECRIKS$, vallgc, astart, asignal

Voir le sous-programme ECRITY$

2.4.2. Lecture d'une chaîne de caractères d'entrée via un télétype rapide

CALL LIREKR$, vallgc, astart, asignal, arecul

- vallgc : nombre de caractères à lire

- astart : voir WAITREP$

- asignal : voir LIREMT$

- arecul : une adresse à laquelle l'éxécution du programme reviendra quand l'opérateur introduit le caractère spécial"CONTROL+X"

2.5. POUR UN MODEM (300 bauds ou 1200 bauds)

2.5.1. Assurance du silence de la ligne de transmission

CALL CALMER$

Ce sous-programme a pour but de débloquer (s'il est nécessaire) la ligne de transmission VARIAN-UNIVAC et d'éviter les bruits sur cette ligne avant d'ouvrir une session conversationnelle entre 2 machines.

Ce sous-programme utilise en plus du registre A, le registre B.

2.5.2. Dialogue entre le VARIAN et l'UNIVAC

Un dialogue se réalise dans l'ordre suivant : le VARIAN envoie une chaîne de caractères, terminée par "CR", à l'UNIVAC, et ensuite, le VARIAN reçoit les informations venant de l'UNIVAC.

CALL DIALOG$, astenvoi, acompreçu, astreçu,
acr1 , acr2, aflag, asignal

- astenvoi : adresse du début de la chaîne à émettre
- acompreçu : adresse d'un compteur contenant le nombre de caractères reçus de l'UNIVAC (non compris les échos des caractères envoyés par VARIAN)
- astreçu : adresse du début de la zone mémoire réservée pour les caractères reçus de l'UNIVAC.
- acr1 : adresse d'un mot contenant la position du caractère CR qui se trouve avant le dernier caractère CR dans la chaîne de caractères reçus (figure A.2)
- acr2 : adresse d'un mot contenant la position du dernier caractère CR dans la chaîne de caractères reçus
- aflag : adresse d'un mot contenant le code de retour

  Code= 0 : fin normale
  Code= 1 : mauvaise réception de l'UNIVAC
  Code= 2 : ligne de transmission coupée
  Code= 3 : erreur de parité

- asignal : voir LIREMT$

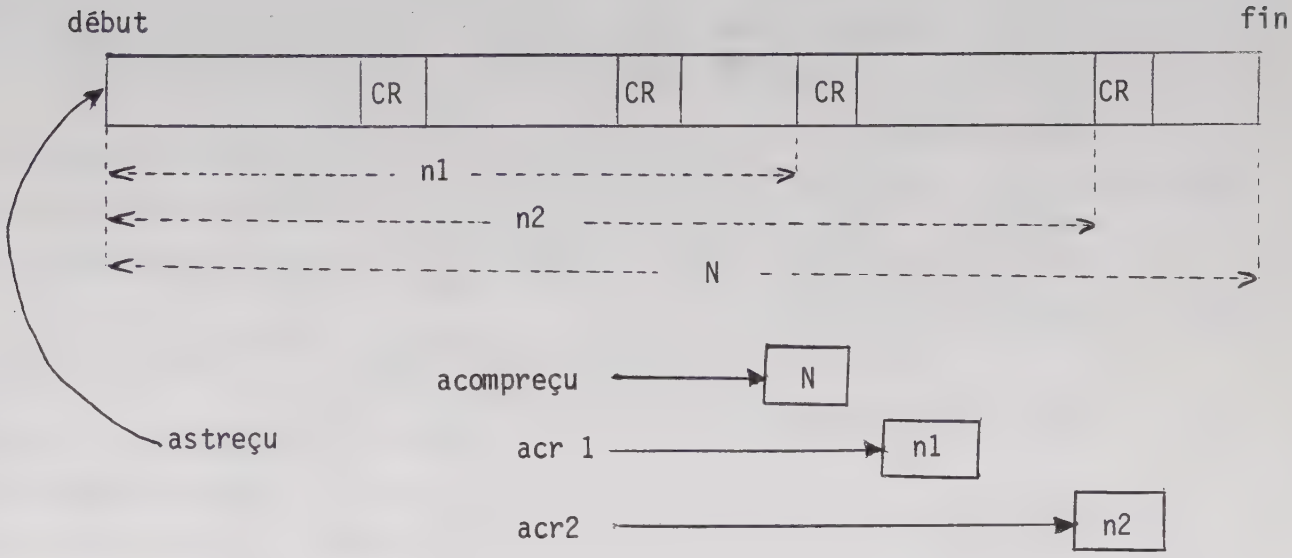


Figure A.2. Chaîne de caractères reçus

2.6. POUR LE DISQUE

Les informations sur disque, quelles qu'elles soient, sont organisées en fichiers. Chaque fichier a dans le répertoire du disque un descripteur contenant les renseignements nécessaires pour toutes les opérations manipulées sur ce fichier.

2.6.1. Ouverture d'un fichier

Ce sous-programme a pour but de demander l'accès à un fichier. Si cette demande est admise, un bloc de contrôle du fichier sera créé en mémoire. Ce sous-programme a besoin des registres A, B et X.

CALL OPENDF$, anom, abloc, valaction

- anom : adresse du début de la zone mémoire (4 mots) contenant le nom du fichier à ouvrir.
- abloc : adresse du début de la zone mémoire (20 mots) réservée pour un bloc de contrôle.
- valaction : action demandée de réaliser sur le fichier.
  valaction = 0 : en lecture seulement
  valaction = 1 : en écriture et lecture

Le code de retour est dans le registre A :

Code = 0 : fin normale, le bloc de contrôle du fichier a été formé en mémoire.

Code = -1 : mauvais fonctionnement du disque (matériel)

Code = 1 : le fichier demandé n'existe pas sur disque

Si la demande d'ouverture d'un fichier n'est pas admise à cause d'un conflit avec un autre programme, l'éxécution du programme demandant l'ouverture sera suspendue.

Le bloc de contrôle d'un fichier sert à gérer les opérations d'entrée-sortie du fichier. Il comporte le contenu d'un descripteur et les informations créées provisoirement pour la méthode d'accès séquentielle pendant la lecture et l'écriture de fichier.

Chaque bloc de contrôle a une longueur de 20 mots et permet de savoir les renseignements montrés dans la figure A.3.

Dans le cas où le code de retour n'est pas égal à 0, le bloc de contrôle n'est pas constitué, mais le contenu de la zone de mémoire réservée pour le bloc de contrôle est modifié pour empêcher les opérations d'entrée-sortie sur ce fichier non-ouvert.

0) Nom de fichier	4) nombre de mots réservés	5)	6)::: pointeur d' écriture	7)* longueur du corps de prog.	8)::: longueur lue actuelle	9) longueur écrite au total

10)::: pointeur de lecture	11)* début de l'éxécution de prog.	12)::: longueur écrite actuelle	13) nombre de secteurs réservés	14)* date	15) début de l'espace réservé	16)* heure	17)* déclaration des unités	18) possesseur du fichier	19) action à réaliser

Figure A.3. Bloc de contrôle

* : seulement pour les fichiers programmes

::: : informations créées provisoirement pour la méthode d'accès séquentielle

2.6.2. Ecriture séquentielle d'un fichier sur disque

L'écriture se réalise au fur et à mesure à partir du début de fichier. La première fois de l'écriture après l'ouverture de fichier commence au début

de fichier. Les fois suivantes réalisent l'écriture sur les fractions qui suivent les fractions précédentes dans l'espace réservé pour le fichier.

Ce sous-programme a besoin des registres A et B.

CALL WRITEDK$, astart, vallg, abloc

- astart : adresse du début de la zone mémoire contenant les informations à écrire sur disque
- vallg : nombre de mots à écrire (multiple de 120)
- abloc : adresse du début de la zone mémoire contenant le bloc de contrôle du fichier

Le code de retour est dans le registre A :

Code = 0 : fin normale
Code = -1 : mauvais fonctionnement du disque (matériel)
Code = 1 : refus dû à un mauvais bloc de contrôle
Code = 2 : refus dû à un dépassement de l'espace réservé pour le fichier sur disque.

2.6.3. Lecture séquentielle d'un fichier sur disque

La lecture se réalise au fur et à mesure à partir du début de fichier. La première fois de la lecture après l'ouverture de fichier commence au début de fichier. Les fois suivantes réalisent la lecture sur les fractions qui suivent les fractions lues précédemment dans la partie réellement écrite de fichier.

Ce sous-programme a besoin des registres A et B.

CALL READDK$, astart, vallg, abloc

- astart : adresse du début de la zone mémoire recevant les informations lues.
- vallg : nombre de mots à lire
- abloc : voir WRITEDK$

Le code de retour est dans le registre A :

Code = 0, -1, 1 : voir WRITEDK$
Code = 2 : fin de la partie réellement écrite de fichier atteinte avant la lecture complète du nombre de mots demandé.

2.6.4. Ecriture et lecture avec accès direct

Ce sont 2 sous-programmes qui ont pour but de réécrire ou de lire une fraction d'informations d'un fichier, choisie à volonté pour chaque fois. La position de cette fraction est définie par rapport au début du fichier en

unité de secteur. Le début du fichier correspond à la position 0.

Ces sous-programmes ont besoin des registres A et B.

Pour écriture : CALL WDKAD$, astart, vallg, abloc, valpot

Pour lecture : CALL RDKAD$, astart, vallg, abloc, valpot

- astart : adresse du début de la zone mémoire contenant les informations à échanger
- vallg : nombre de mot à lire ou à écrire
- abloc : voir WRITEDK$
- valpot : nombre de secteurs à sauter par rapport au début de fichier

Le code de retour est dans le registre A (voir WRITEDK$)

2.6.5. Fermeture d'un fichier

Ce sous-programme a pour but de libérer un fichier de l'occupation d'u possesseur, et aussi de changer la longueur de sa partie réellement écrite après des écritures séquentielles consécutives.

Il a besoin des registres A, B et X.

CALL CLOSEDK$, abloc

- abloc : voir WRITEDK$

Le code de retour est dans le registre A :

Code = 0 : fin normale, le descripteur de fichier sur disque a été modifié en ce qui concerne l'état de fichier et éventuellement la longueur de la partie réellement écrite. Après quoi, le contenu du bloc de contrôle en mémoire est modifié afin de ne pas être utilisable pour les opérations d'entrée-sortie sur le fichier concerné.

Code = -1 et 1 : voir OPENDF$

Code = 2 : refus dû à un mauvais bloc de contrôle

* Attention -

- Pour des raisons de structure physique, l'écriture ou la lecture sur disque demandée par un appel à un sous-programme commence obligatoirement sur un début de secteur (la longueur de chaque secteur est de 120 mots). Par conséquent, le programmeur doit soigneusement définir la valeur du paramètre "vallg" pour relire exactement les informations écrites.

- Le code de retour qui signale le mauvais fonctionnement du disque (matériel) n'est mis dans le registre A qu'après plusieurs essais de relancer l'opération par sous-programme sans succès.

3 - QUELQUES NOUVEAUX SOUS-PROGRAMMES

Ci-dessous, on présentera quelques nouveaux sous-programmes nécessaires dans le contexte du nouveau système.

3.1. Attente de la fin totale d'une opération d'entrée-sortie.

CALL WAI$$, asignal

- asignal : adresse d'un mot dont le contenu non nul signale la fin totale de l'opération d'entrée-sortie attendue (même paramètre "asignal" dans un appel à un sous-programme d'entrée-sortie précédent)

Ce sous-programme oblige un programme à se mettre en attente d'un évènement indispensable à sa continuation. Son utilisation est obligatoire entre une lecture des informations sur les unités périphériques (microdensitomètre, bande,imprimante à clavier ou modem) et le traitement de ces informations.

3.2. Test de l'état des clés logiques

Ce sous-programme a pour but de remplacer les instructions concernant les clés physiques sur le panneau du calculateur. Maintenant, la mise manuelle des clés physiques est remplacée par la réalisation d'une commande qui met une valeur choisie par l'opérateur dans un mot appelé "clés logiques" appartenant au programme. Ce mot se trouve dans la zone du système en mémoire.

Ce sous-programme teste le contenu du mot "clés logiques".

CALL KEY$, valposit

- valposit : une valeur entière qui est en réalité une suite de bits dont les bits valant 1 indiquent la position des bits à tester dans le mot "clés logiques".

Le code de retour est dans le registre A :

. Code = 0 : tous les bits testés valent 0

. Code ≠ 0 : au moins un bit parmi les bits testés vaut 1

4 - MISE AU POINT D'UN PROGRAMME

Les phases d'assemblage, d'édition de liens et de transformation en module chargeable sur disque ou sur une bande magnétique de l'ordinateur UNIVAC se réalise en utilisant la bibliothèque du nouveau système du VARIAN. Cette bibliothèque est un fichier sur disque de l'UNIVAC contenant le module système, le programme "loader", les procédures et les sous-programmes compatibles avec le nouveau système.

Un module chargeable a 3 parties : l'en-tête ayant une longueur de 20 mots, le corps du programme et le descripteur des adresses translatables.

La phase de transformation en module chargeable permet de connaître les renseignements dans l'en-tête du module (figure A.4.) . Ces informations sortent sur un terminal de l'UNIVAC.

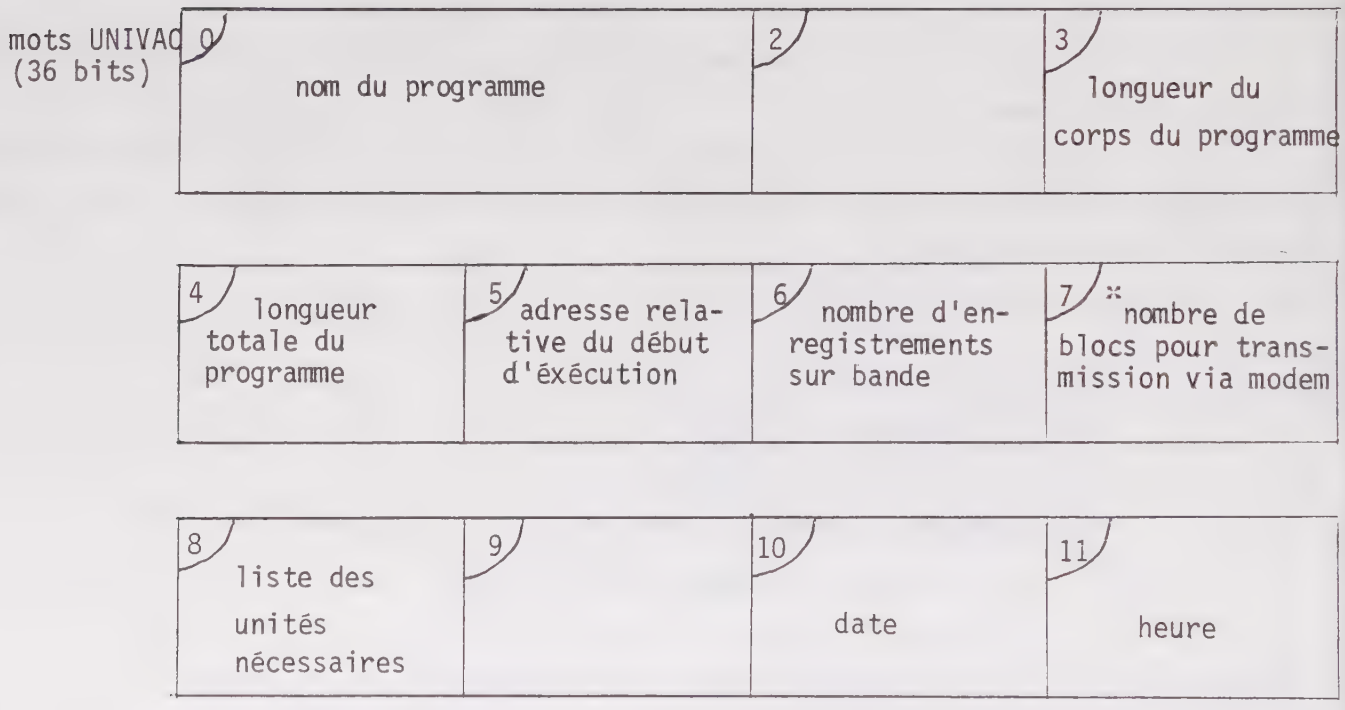


Figure A.4. - L'en-tête d'un module chargeable

(* : seulement pour un module chargeable sur disque UNIVAC)

Dans le cas d'un module sur bande, l'en-tête est séparée des autres parties par une "fin de fichier". Le corps du programme et le descripteur des adresses translatables sont considérés comme un fichier unique sur bande avec des enregistrements de longueur fixe 480 mots (i.e. 4 secteurs) sauf le dernier enregistrement dont la longueur doit satisfaire l'inégalité suivante :

$$4 \times 120 \text{ mots} \leqslant \text{longueur} < 8 \times 120 \text{ mots}$$

(voir la figure A.5)

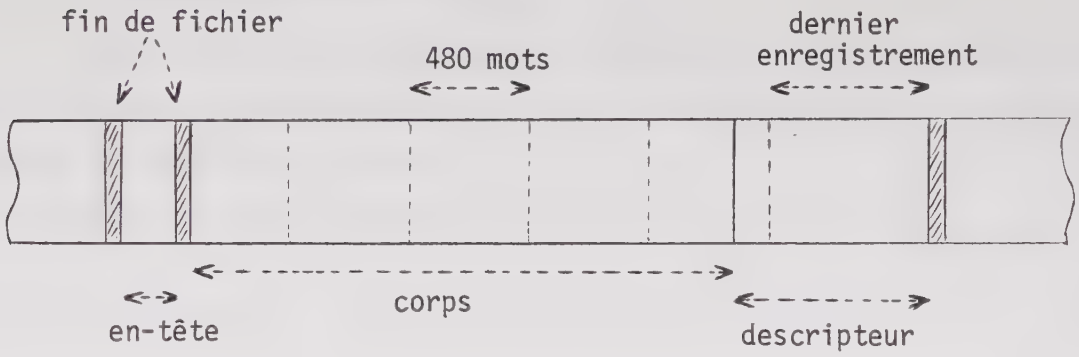


Figure A.5.- Structure d'un module chargeable sur bande

La transmission d'un module chargeable via une ligne de transmission ou le chargement d'un module à partir d'une bande au disque du VARIAN est réalisé par un programme utilitaire. Avant de le faire, il faut créer un fichier sur disque du VARIAN pour contenir ce module. Ceci est fait par un programme utilitaire qui demande des informations nécessaires comme le nom du programme, la longueur du fichier qui doit être supérieure ou égale à la somme de la longueur du corps du programme et la longueur du descripteur des adresses translatables. Dans le cas de chargement à partir de bande au disque, la création des fichiers sur disque peut aussi être faite automatiquement selon les renseignements dans l'en-tête des fichiers sur bande.

A N N E X E 2

COMMANDES OPERATEUR

Toutes les commandes et leur informations supplémentaires commencent par "1-" et se terminent par "return".

Avant d'introduire une commande concernant la bande ou le microdensitomètre, il faut s'assurer que cette unité périphérique soit libre (c'est-à-dire qu 'aucun programme ne s'en sert).

1 - Lister le nom des fichiers sur disque :

1- CD

2 - Lister le nom des fichiers sur une bande de sauvegarde :

1- CT

3 - Charger un programme en mémoire à partir de disque :

1- LD

puis 1- nom du programme

4 - Charger un programme en mémoire à partir d'une bande :

1- LT

puis 1- nom du programme

5 - Lister le nom et le numéro des programmes en mémoire :

1- OB

6 - Supprimer l'éxécution d'un programme :

1- SP

puis 1- numéro du programme (2, 3)

7 - Ecrire une fin de fichier sur une bande :

1- MK

8 - Rebobiner la bande :

1- RW

9 - Remettre à la position 0 du chariot du microdensitomètre :

1- PZ

10 - Mettre des clés logiques pour un programme :

1- Kn

puis 1- v

où : n = numéro du programme (2 ou 3)

v = 1 caractère de chiffre de 0 à 7

11 - Reprendre inconditionnellement le contrôle pour opérateur.

Cette commande permet de supprimer l'éxécution de la commande en cours, mais elle n'a aucune influence sur le programme qui est en train de s 'éxécuter.

Elle ne commence pas par "1-". Sa forme est la suivante :

&& (return)

A N N E X E 3

MESSAGES DU SYSTEME

Cette annexe présente les messages du système et leur signification dans chaque contexte spécifique.

SYSTEM <date> - Le système commence à fonctionner.
"Date" montre la date de création du système.

C : Le calculateur est prêt à recevoir une commande de l'opérateur.

→ E0 La commande introduite est incorrecte.

→ E1 Le chargement d'un programme est suspendu à cause de mauvais fonctionnement de disque ou d'erreur parité de bande.

→ E2 Le chargement d'un programme est suspendu à cause d'erreur de mot de contrôle (disque) ou d'erreur de longueur (bande).

→ E3 Le chargement d'un programme est suspendu à cause d'erreur de positionnement de disque.

→ E4 Le chargement d'un programme est refusé à cause du conflit de ressource

→ FF La recherche des fichiers arrive la fin du support. Dans le contexte de chargement d'un programme, cela signifie que le programme à charger n'existe pas sur le support.

→ 00 Le programme à supprimer n'existe pas en mémoire.

→ KS Le chargement d'un programme est refusé à cause du manque d'une imprimante à clavier.

→ LN Le chargement d'un programme est refusé à cause du manque d'un modem.

→ PH Le chargement d'un programme ou l'éxécution d'une commande concernant microdensitomètre est refusé car celui-ci n'est pas prêt.

PH Le programme en cours attend l'état prêt du microdensitomètre.

→ MT Le chargement d'un programme ou l'éxécution d'une commande concernant le dérouleur de bande est refusé car celui-ci n'est pas prêt.

MT Le programme en cours attend l'état "on-line" du dérouleur de bande ou bien il attend une bande en écriture.

WRITE Le programme à charger nécessite une bande en écriture.

<n>.<nom du program> - Le programme chargé en mémoire porte le numéro n.

::: END : <nom du programme> - Le programme se termine.

:::// D : <nom du programme> - L'éxécution d'un programme est suspendu par le système.

Les raisons sont diverses :

- Conflit de fichier sur disque (avec la sortie à l'avance du message : CONFLIT DK)
- Position du tambour du microdensitomètre mal réglée (avec message OVERRUN
- Boucle infinie dans programme.

: COUPURE COURANT - Une chute de tension a eu lieu. Le fonctionnement du calculateur s'arrête.

BIBLIOGRAPHIE

1 - ARSAC J.

Les systèmes de conduite des ordinateurs. Dunod Université, 1968.

2 - DU ROSCOÄT J.

Principes et problèmes d'un système d'exploitation d'ordinateur - Masson et Cie, 1972.

3 - PETERSON J.L.

Petri Nets - Computing Surveys, Septembre 1977.

4 - PURSER W.F.C., JENNINGS D.M.

The Design of a Real - Time Operating system for a Minicomputer. Part One - Software Pratice and Experience, Vol. 5, No 2, April- June 1975.

5 - TAUPIN D., ZELWER C., NOGUES J., GUINIER A.

Le microdensitomètre. Le courrier du CNRS, n°15, Janvier 1975.

6 - VARIAN 620 - 100 serie - Computer handbook - Varian data machines, 1972.